## Re-design of compositional systems

Wijngaards, N.J.E.

1999

# Re-design of compositional systems

SIKS Dissertation Series No. 99-6

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems.

This research has been supported by the Netherlands Organisation for Scientific Research (NWO).

ISBN: 90-9012977-4

VRIJE UNIVERSITEIT

# Re-design of compositional systems

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan
de Vrije Universiteit te Amsterdam,
op gezag van de rector magnificus
prof.dr. T. Sminia,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
wiskunde en informatica
van de faculteit der exacte wetenschappen
op donderdag 30 september 1999 om 13.45 uur
in het hoofdgebouw van de universiteit,
De Boelelaan 1105

door

Niek Jaap Edwin Wijngaards

geboren te Eindhoven

Promotor: prof.dr. J. Treur
Copromotor: dr. F.M.T. Brazier

# Acknowledgements

Research for a PhD thesis not only takes time, but also depends on the help of many persons.

First of all I would like to thank Jan Treur for being my promotor, and Frances Brazier for being my co-promotor. Their guidance, good humour, and visions have greatly contributed to my development as a researcher. The availability of tea, open doors, and attention has made our co-operations a genuine pleasure.

Furthermore, I would like to thank the reading committee (Hans Akkermans, Frank Dignum, Ashok Goel, Frank van Harmelen, and Bob Wielinga) for reading and commenting on an earlier version of this thesis.

The members of the REVISE project (Hans Akkermans, Frances Brazier, Frank van Harmelen, Jan Treur, Bob Wielinga, and Nel Wognum) contributed to my understanding of different approaches to modelling design activities. The co-operation and discussions with the PhD-students in the REVISE project, Anita Pos, Remco Straatman and Bas van Eldonk, have been constructive and valuable. Discussions with members of the BBC ("building-block club") have always been stimulating and fun. The members were Manfred Aben, Richard Benjamins, John van den Elst, Dieter Fensel, Frank van Harmelen, Mark Sloof, Remco Straatman, and Annette ten Teije.

The colleagues at the Department of Artificial Intelligence at the Vrije Universiteit Amsterdam have been supportive of each other. Mark Sloof, Frederik Jan Jüngen, Pascal van Eck, Joeri Engelfriet, Wouter Wijngaards, Catholijn Jonker, and Rineke Verbrugge have contributed to the pleasant work environment. Collaborating with my co-authors, especially Frances Brazier, Catholijn Jonker, Pieter van Langen, and Jan Treur, has been educational and enjoyable, which is in my opinion the right combination for doing research. Special thanks go to Pieter van Langen, with whom I have extensively collaborated during my research. His work on the anatomy of design is closely related to my work on re-design. Our lively discussions of generic, and not-so-generic, aspects of design and re-design have boosted our mutual understanding of intricacies of (re-)design processes.

During my year as a Post Doctoral Fellow at the Software Engineering Research Network at the University of Calgary, I continued to work on my thesis in my spare time. Combining all the work with living abroad was greatly facilitated by the presence of friendly colleagues: Mildred Shaw, Rob Kremer, Roberto Flores Mendes, Michele Jacobsen, Frank Deur, Frank Maurer, Brian Gaines, Dana Herlea, Andy Kremer, Camille Sinanan, the CAG -group including Mihaela Ulieru and Douglas Norrie, and others. My host family in Calgary has been very important to me: again, thanks for all your kindness and support.

The conferences and workshops I visited have contributed to my understanding of Knowledge Engineering. Hereby I acknowledge the participants at the KAW, the EKAW, and the KEML workshops for their presentations, questions, and discussions.

Friends have been very important to me during these years. Special thanks go to Karin Ooiman, Huib van Nieuwenhuijze, and Mark Sloof, who even visited me during my stay in Canada.

Last, but not least, I wish to thank my parents and brother for their support during these years of research. Especially I wish to thank my partner Karin Lassche, for all her kind support during these laborious years.

*Niek*

# Table of Contents

## Part V



## Part VI

Part **I**

# Introduction and Research Perspective

In this part an introduction is given to the research presented in this thesis and its context. In Chapter 1 the processes of design and re-design are introduced and compositionality as a structuring principle is explained. In Chapter 2 an outline of the research context is presented: the central research theme is identified and demarcated. More specific desiderata are formulated. Relevant literature for this thesis is outlined in Chapter 3. This includes a discussion on design, and, in particular, software design processes, structuring principles of software, and an introduction to two domains of application.

# 1 Introduction

Re-design of compositional systems involves design processes, compositional systems, and the combination of design and compositional systems. *Design* is an activity common to humans; a fact testified by the presence of the artefacts surrounding us. Often new artefacts are designed on the basis of existing artefacts, an activity named re-design. The activity of design is a complex activity and approaches are sought to structure this complex activity; one approach is to distinguish *components* within an artefact.

Many artefacts in the real world are designed by distinguishing components; the components themselves may, in turn, be constructed from (smaller) components. A table, for example, may have four legs and a top: that makes five less complex components, as illustrated in Figure 1.1. Each leg of this table consists of a long thin piece of material (e.g., wood) with a bracket on one end (e.g., metal): that makes two simpler components, for each of the four legs. The connections between the components are of importance, for example, the legs of the table can be connected to the top in several ways (e.g., screws or woodwork), each method has its own strengths and weaknesses. The approach of identifying entities called components and defining a way to combine components together to make a new component, is termed *compositional design*.

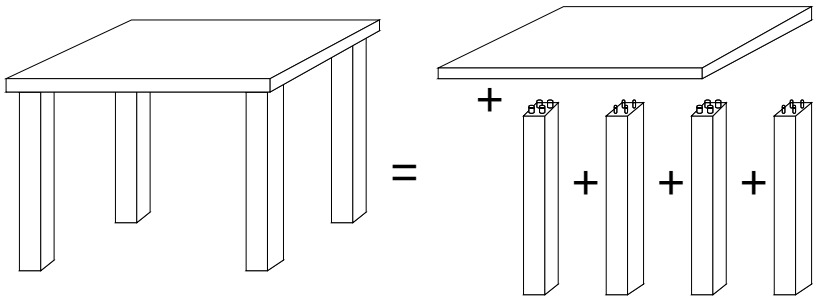


**Figure 1.1** An example of compositionality: a table can be viewed as composed of a top and its legs.

The notion of components can easily be found in products manufactured by modern industry: more and more component-based artefacts are currently being manufactured, ranging from cars to aircraft, from abacuses to computers. Assembly lines in factories are based on the principle of compositionality: components are combined to form an end product.

Within the software industry, compositionality is also employed; at the very least to divide software into manageable portions, but also to create libraries of software components that can be used during design. The design of software has not been fully automated. Modelling relevant information and identifying relevant processes is one of the most difficult activities in software design: it involves an understanding of complex structures in real world domains. As in the design of physical objects, compositional structures of software can provide additional structure for processes of design of software.

Within the design of software a distinction can be made between processes and knowledge. The notion of compositionality can be applied to both. Compositionality of processes provides a means for ‘process hiding’ and compositionality of knowledge provides a means for ‘knowledge hiding’. Compositionality of both processes and knowledge can be combined within one compositional system.

Re-design is closely related to design: a process of re-design can be regarded as a process of design, starting with an initial design. In Figure 1.2 a simplified view on re-design is depicted.

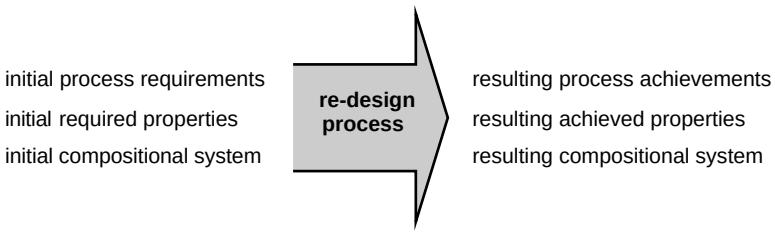


**Figure 1.2** Simplified view on input and output information of a re-design process

The input and output information in this simplified view on a process of re-design can be described as follows:

The input of a re-design process consists of:

- requirements on a re-design process (e.g., a new word processor must be produced within one week),
- required properties of a compositional structure (e.g., new behaviour is required: faster production of correct English text), and
- an initial description of a compositional system (e.g., a word processor).

The output of a re-design process consists of:

- resulting process achievements (e.g., four days were needed for this re-design process),
- resulting achieved properties (e.g., modification of an initially required property: automatic correction of spelling errors), and
- a resulting description of a compositional system (e.g., a new word processor with automatic spelling correction).

The initial requirements on the *compositional system* (i.e., required properties of the compositional system) are often modified during the re-design process. Requirements on the re-design *process* influence strategies applied within the re-design process: clearly a different result can be obtained if, for example, much more time or financing is available.

## The structure of the thesis

In the next chapter, Chapter 2, the research context is outlined: the overall research theme is identified and demarcated.

Relevant literature on design processes (of software) and structuring principles (of software) is briefly discussed in Chapter 3. Chapter 3 also includes an introduction to two domains of application: diagnostic reasoning systems and multi-agent systems. Each domain of application is used in a later chapter to illustrate the re-design of a compositional system.

In Chapter 4 a representation for a compositional system for knowledge-intensive systems is presented. Two examples illustrate the use of this compositional system: an example diagnostic reasoning system and an example multi-agent system.

In Chapter 5 properties of compositional systems for knowledge-intensive systems are described. These properties provide a means to describe the functionality of a compositional system. Relevant research is outlined, as well as different approaches to obtaining such properties. Properties and knowledge on properties are identified for the two example compositional systems: diagnostic reasoning and multi-agent systems.

As processes of re-design are an inherent part of processes of design, first a generic model for processes of design is described in Chapter 6.

In Chapters 7, 8, and 9 a refinement of the generic design model is described. This model contains additional detail for the re-design of compositional systems.

In Chapter 10 a model for a design agent is described, based on a model for re-design (Chapters 7, 8, and 9) and an agent model (Chapter 4). This design agent is capable of re-design of compositional systems, in this case re-design of a multi-agent system, as described in Chapter 12.

In Chapters 11 and 12 two illustrations of the applicability of the re-design model of Chapter 8 are given. The re-design of a diagnostic reasoning system is described in Chapter 11, and the re-design of a multi-agent system is described in Chapter 12.

A discussion of the results of the research described in this thesis, in addition to ideas for future research, is presented in Chapter 13.

In Appendix A additional details of the generic model of design (see Chapter 6) are described.

In Appendix B additional details of the model for re-design of compositional systems (see Chapters 7, 8, and 9) are described.

# 2 Research Context

The central research theme of this thesis combines the compositional structures of artefacts and processes of re-design. This research theme is demarcated by a focus on a specific range of systems: knowledge-intensive compositional systems.

*How can a compositional structure be used to re-design knowledge-intensive systems?*

A compositional structure can be used in two ways: to structure the *process* of re-design (a knowledge-intensive process), and to structure the *design object* (a knowledge-intensive system).

In the next two sections the overall research theme is demarcated in more detailed desiderata on processes of re-design and knowledge-intensive compositional systems.

## 2.1 Research themes concerning a process of design

Re-design is an inherent part of almost every process of design. Within each design process intermediate descriptions of design objects are analysed and modified on the basis of a set of qualified requirements and design process objectives. Processes of re-design are often characterised as design processes that start with an initial description of a design object that needs to be modified. Several models and theories exist for processes of design, as discussed in Chapter 3 and Chapter 6.

Design is a knowledge-intensive task: a designer reasons about descriptions of artefacts, reasons about requirements, and employs strategies to structure the design process. Models of how human designers approach design are often based on analyses of design tasks, and designers' approaches (e.g., Akin, 1978; Schön, 1983; Pahl and Beitz, 1984; Brown and Chandrasekaran, 1989; Chandrasekaran, 1990; Smithers, Corne and Ross, 1994). To understand, describe and model a process of design, knowledge level (Newell, 1982) theories of design are needed (Smithers, 1996), one of which is proposed in (Brazier, Langen and Treur, 1996).

Design can be viewed as a process of the creation of a set of requirements and a design object description that satisfies these requirements, on the basis of initial requirements and preferences specified by agents, and libraries of existing designs, while adhering to design process objectives, see Figure 2.1. A generic model of design described on the basis of these concepts and sub-processes is proposed in (Brazier, Langen, Ruttkay and Treur, 1994) and is described in more detail in Chapter 6. An application of this generic model of design is described in (Brazier, Langen, Treur, Wijngaards and Willems, 1996).

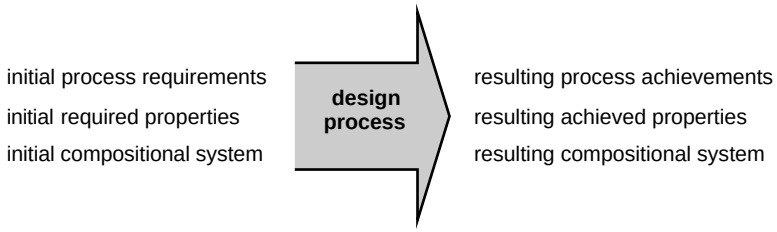


**Figure 2.1** Simplified view on input and output information of a design process.

Modelling design entails modelling the domain (i.e., the world of interest), the requirements for each of the parties involved, the design objects, and the design process. A description of a design object is needed (e.g., a blueprint of an artefact and assembly instructions).

Requirements (and preferences over requirements) state which properties and structures should guide the design process. Objectives can be given for the design process; e.g., resource limitations such as limited time and funding. An overview of possible input and output of a process of design, in retrospect, is given in Table 2.1.

| *Concepts within a process of design* | *Input of design process* | *Output of design process: scenario 1* | *Output of design process: scenario 2* |
|---|---|---|---|
| *Design object description* | Possibly inconsistent, possibly incomplete, design object description. | Resulting design object description satisfying resulting set of qualified requirements. | No resulting design object description. |
| *Qualified requirements* | Possibly conflicting, possibly unrealisable, set of qualified requirements. | Non-conflicting, realisable, resulting set of qualified requirements. | Conflicting, non-realisable set of qualified requirements with indications of problems. |
| *Design process objectives* | Possibly conflicting, possibly unrealisable, set of design process objectives. | Non-conflicting, realised set of design process objectives. | Non-conflicting, non-realised set of design process objectives. |

**Table 2.1** Overview of input and possible outputs of a design process.

A design process is capable of producing intermediate results, e.g., for negotiation of (qualifications of) requirements. Two possible outcomes, given an initial situation, are described in Table 2.1. In the initial situation an initial design object description is provided, which may be inconsistent (i.e., conflicting with domain theories) or incomplete (i.e., partial). An initial set of qualified requirements is provided, which may contain conflicting requirements or unrealisable requirements. Initial design process objectives are provided which may also by conflicting, or unrealisable.

- In the first scenario the design process terminates with: a set of agreed qualified requirements, a design object description which satisfies this set of qualified requirements, and achievements of the design process objectives.
- In the second scenario, the design process terminates with: no design object description, a set of qualified requirements with conflicts and unrealisable requirements and indications of where problems reside, and achievements of the design process objectives.

A process of design may be entirely automated, or it may be an interactive process. An interactive design process supports collaboration between systems (automated or human): decisions regarding the selection of appropriate strategies, modifications to sets of qualified requirements, and modifications to design object descriptions can be made in a co-operative manner. In such situations it is very important that not only shared concepts are used to describe design object descriptions, but also a shared language exists with which requirements can be communicated (Fischer and Lemke, 1988). Users may have the role of manipulating the requirements while interacting with a design process, which can be modelled by means of structured dialogue (Murray and Sheppard, 1988; Forbus, 1988). A shared model (Brazier, Treur and Wijngaards, 1996*b*; and for an extended version: Brazier, Jonker, Treur and Wijngaards, 1999*b*) of a process of design can be employed to structure the interaction and communication between a design support system and its expert user, allowing the user to exert greater control of, and influence on, the design process.

A process of design as described in this chapter is comprehensive; in some cases the domain of application of the design process can be more restricted. For example, in a given situation the manipulation of sets of qualified requirements and co-ordination of the design process may be considered superfluous, and only the manipulation of the design object description may be modelled and specified. Usually a specific strategy is then ‘built into’ the manipulation of the design object description.

The processes of design and manufacturing are closely related. Results of a design process, e.g., a design object description, are input to a manufacturing process. Results and

experiences obtained from a manufacturing process may, in turn, influence a design process, for example, to re-design the design object description to meet additional requirements.

The list of desired properties of a design model, to be used as a basis for the research presented in this thesis, includes:

- Explicit distinction between the manipulation of design object descriptions, manipulation of sets of qualified requirements, and co-ordination of the design process.
- Explicit representation and manipulation of design process objectives. Directives for the design process on resource allocation (e.g., time, money) and other strategical aspects are explicitly represented and manipulated.
- Explicit representation and manipulation of (sets of) qualified requirements. Requirements and their qualifications are explicitly represented and manipulated.
- Explicit representation and manipulation of design object descriptions. Descriptions of design objects are explicitly represented and manipulated.

The model for design processes, described in (Brazier, Langen, Ruttkay and Treur, 1994) and described in more detail in (Brazier, Langen and Treur, 1999), is adopted as the model for design processes in this thesis. It fulfills the desired properties listed above, as can be deduced from the description in Chapter 6. The model provides a structure that indicates which types of knowledge structures and processes need to be specialised and thus acquired (model-based knowledge-acquisition). Acquisition is not limited to alter existing structures but requires further investigation.

In this thesis these principles, together with the adopted model of design, are employed to construct a model for re-design of compositional systems. To be more precise, the model for re-design of compositional systems is a *refinement* of the model of design. The term ‘design’ is used to denote both design and re-design. Desiderata (i.e., desired properties) on the re-design of compositional systems are formulated below.

**dr1** *Representation of a knowledge-intensive system as a design object description.* The representation format has been inherited from the generic design model; an ontology to express the structure and characteristics of knowledge-intensive systems is required.

**dr2** *Representation of qualified requirements on compositional systems.* The representation format has been inherited from the generic design model; an ontology to express the structure and characteristics of requirements is required.

**dr3** *Model of the manipulation of compositional system structures.* Knowledge on the relationship between requirements on a compositional structure for knowledge-intensive systems and possible modifications to a compositional structure is explicitly specified, e.g., deductive refinement knowledge (to establish properties), and knowledge which relates (required) properties to possible structures.

**dr4** *Model of the manipulation of requirements on compositional systems.* Knowledge of relationships between qualified requirements (i.e., properties of compositional structures for knowledge-intensive systems) is explicitly specified, e.g., knowledge on refinements of requirements, and knowledge on properties of requirements for assessment.

**dr5** *Knowledge on the co-ordination of the re-design process.* Knowledge on design strategies for the design of knowledge-intensive systems has to be included.

**dr6** *Model of integration of design and realisation.* The processes of design and realisation (e.g., manufacturing or implementation) can be integrated to the extent that design object descriptions resulting from a design process can be implemented by a realisation process, from which feedback can be used to guide the design process.

**dr7** *Model of self-modification.* A system modifying itself, e.g., adapting itself to a new environment or learning new skills, can be realised by a specific integration of a design process and a realisation process.

## 2.2 Research themes concerning the objects of design: structures of knowledge-intensive systems

Designing knowledge-intensive systems is a process in which support plays an important role. Appropriate computer support in designing requires that a formal, unambiguous, representation language is used to describe a knowledge-intensive system, and formal semantics are attributed to such a description. Computer support in realisation requires that dynamic aspects of a knowledge-intensive system are specified such that fully automated operationalisation of a knowledge-intensive system becomes possible.

Another requirement for automated design of knowledge-intensive systems is an appropriate (internal) structure of the knowledge-intensive system. A compositional structure provides a means by which a design process can be guided. By distinguishing compositionality of processes and compositionality of knowledge, structures are available to guide the process of design.

The above considerations have been formulated as the following desiderata on descriptions of knowledge-intensive systems.

**ds1** *Unambiguous, formal, representation language*. A language is needed with which a compositional structure is described in an unambiguous and formal manner.

**ds2** *Formal semantics.* A formal semantics can be attributed to the compositional structure.

**ds3** *Explicit representation of both static and dynamic properties*. A language and ontology is needed in which an ontology of properties (both static and dynamic) of (part of) a compositional structure can be expressed.

**ds4** *Operationalisation.* A system represented in a formal language can be (automatically) operationalised (e.g., into prototype systems), thereby facilitating the testing of systems (compared to testing a system by first manually implementing the system in another language).

**ds5** *Compositionality as a structuring principle.* Compositionality of both processes and knowledge provides structure in a knowledge-intensive system, together with an explicit relation between a process composition and a knowledge composition.

# 3 Relevant Literature

In this chapter relevant literature is discussed. In Section 3.1 literature on structuring principles for design processes is addressed; in Section 3.2 literature on structuring principles for knowledge-intensive software systems is described. Example domains of application taken from the literature and used to illustrate the re-design of compositional systems are described in Section 3.3. In Section 3.4 the results are discussed in relation to the research theme of this thesis.

## 3.1 Structuring principles for design processes

Design is most often an activity which involves extensive human expertise. The process of design is generic: design occurs in many areas, including engineering design and software design. In engineering design the focus is on finding a configuration of certain physical elements that, combined in one artefact, perform the required functions (Pahl and Beitz, 1984; Koller, 1985; Alberts, 1993). Similarly, in software design, a configuration of program-components has to be found that, combined into one program (i.e., the artefact), performs the required functions. In both areas (required) function is related to the structure of an artefact. Also, in both areas structured artefacts are employed and properties can be formulated to reflect the functionality or behaviour of an artefact.

This thesis focuses on the design of compositional knowledge-intensive systems. The literature discussed in this section ranges from design in general (of any artefact) to design of software. First models for design processes in general are discussed, after which generic methods and techniques for design are addressed, and finally software design is discussed in relation to Requirements Engineering, Software Engineering and Knowledge Engineering.

### 3.1.1 Models for design processes

A substantial amount of research has focused on defining models of design as a basis for knowledge-based design systems; e.g., (French and Mostow, 1985; Tomiyama and Yoshikawa, 1987; Treur, 1989; Brown and Chandrasekaran, 1989; Chandrasekaran, 1990; Coyne, Rosenman, Radford, Balachandran and Gero, 1990; Gero, 1990; Takeda, Veerkamp, Tomiyama and Yoshikawa, 1990; Alberts, Wognum and Mars, 1992; Tham and Gero, 1992; Vescovi and Iwasaki, 1993; Ohsuga, 1997, Brown and Birmingham, 1997). While modelling of the functionality (or properties) of the design object description is addressed as an important aspect of a process of design (for an overview see Winsor, McCallum, 1994) not many approaches have included actual reasoning about these properties in the context of requirement manipulation. Some of these models recognise manipulation of requirements or strategies as an important part of (re-) design. Relevant literature on models for (re-)design is addressed below.

In (Koller, 1985) the process of design consists of synthesis and selection (or analysis) processes, where the selection (or analysis) process validates results of the synthesis process. In (Pahl and Beitz, 1984) the process of design consists of explanation of the problem description, conceptual design, detailed design, and manufacturing. The need for design theories is recognised (e.g., Dixon, 1989), and resulted in conferences on design theories (Gero, 1996).

Models of processes of design provide a structured description of a process of design. Models of design differ in their underlying formalisations. Models are represented in structures such as blackboard architectures (e.g., Ball and Bauert, 1992); algorithms (e.g., Alberts, Bakker, Deekman and Wognum, 1993), SOAR (Steier, 1991), task models or problem solving methods (Brown and Chandrasekaran, 1989; Brazier, Langen, Ruttkay and Treur, 1994;

Wielinga and Schreiber, 1997), or agent architectures (Dunskus, Grecu, Brown and Berker, 1995; Berker and Brown, 1996; Lander, 1997).

One approach employed to model design tasks, is to design as the application of the problem solving method "Propose & Revise" (Marcus and McDermott, 1989) in which a tentative solution is generated and modified. Propose-and-Revise is based on the problem-solving methods "Propose-Critique-Modify" and "Propose-Verify-Redesign" (Chandrasekaran, 1986; Chandrasekaran, 1990; Goel and Chandrasekaran, 1989). Within the context of parametric design several experiments with Propose-and-Revise have been performed by Zdrahal and Motta (1995).

A perspective on engineering design as a synthesis process is described by (Alberts, 1993). Original requirements and basic generic elements are input of the design process, and final requirements and product descriptions are output of the design process. This perspective on engineering design includes the manipulation of requirements (and the manipulation of a product description) but does not explicitly include objectives on the design process itself.

A model of design proposed by (Ohsuga, 1997) features both the manipulation of a design object description as well as strategic knowledge on the management of this process, as depicted in Figure 3.1 (Ohsuga, 1997, pp. 5). Two kinds of knowledge are identified in this model: knowledge applied directly to the model being designed, and knowledge to guide and control the exploration or search process.

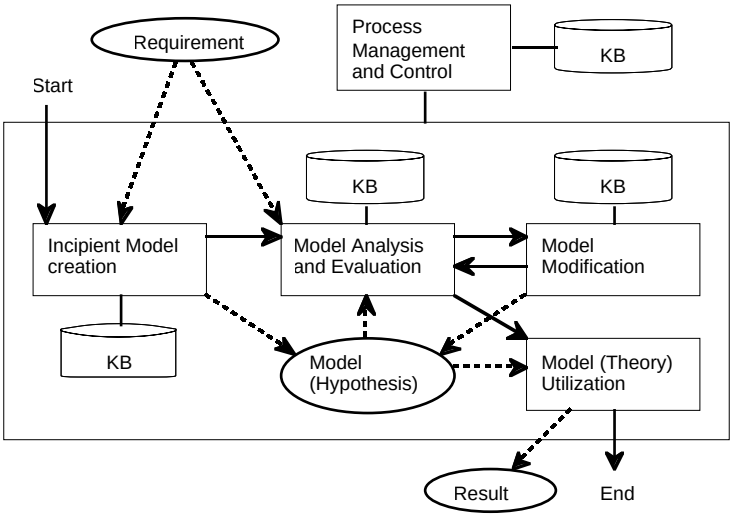


**Figure 3.1** Exploratory problem solving (Ohsuga, 1997, pp. 5)

An extension of this model investigates the manipulation of sets of requirements in interaction with users (Sumi, 1997). An experience-based approach is taken, allowing users to explore the space of requirements. The approach of Ohsuga and Sumi offers a realisation of the high-level desiderata on design described in Chapter 2.

Another model in which both the manipulation of requirements and the manipulation of design object descriptions are discerned is proposed by Smithers (1992). From his viewpoint of design as exploration, both the exploration of possible sets of requirements as well the exploration of possible design object descriptions are explicitly modelled (Smithers and Troxell, 1990).

Models for design processes incorporate ontologies: ontologies for design objects and ontologies for requirements. Ontologies can be briefly characterised as descriptions of concepts in the world. Ontologies for design objects can be shared across domains and incorporated in design processes (Alberts, 1993; Gruber and Olsen, 1994; Borst, Akkermans and Top, 1997). Within the Ontolingua project (Gruber, 1993) an ontology for a process of design has been proposed which is geared towards the representation of design object descriptions.

Likewise, ontologies can be employed to represent requirements. Examples of ontologies that represent requirements (e.g., required properties of compositional systems) are: properties of diagnostic systems (Benjamins, 1993; Cornelissen, Jonker and Treur, 1997), properties of propose-and-revise problem-solving methods (Fensel and Motta, 1998).

### 3.1.2 Generic methods and techniques for design

Within a process of design, methods and techniques are employed for the use of design strategies, design object manipulation, requirement manipulation, and the use of design rationale. These methods and techniques are briefly discussed below.

**Strategies.** Explicit representation of knowledge on strategies within a process of design provides additional structure to the process of design (Rist, 1995). Acquiring, representing and applying strategic knowledge within design processes is a focus of research on its own, e.g., (Gruber, 1990; Strelnikov and Dmitrevich, 1991; Rist, 1995; Ohsuga, 1997; Hori, 1997). Strategic interaction becomes feasible when strategies are explicitly represented, thereby allowing the user of a system greater flexibility and control over the system (Brazier, Treur and Wijngaards, 1996*b*; Brazier, Jonker, Treur and Wijngaards, 1999*b*).

One perspective on a process of design is that of 'exploration of design space', where the design space consists of two subspaces: one for the possible sets of qualified requirements and one subspace for possible design object descriptions. The overall design strategy of the design process determines the exploration strategies (Brazier, Langen, Ruttkay and Treur, 1994; Brazier, Langen and Treur, 1998; Logan and Smithers, 1992; Löckenhof and Messer, 1994).

**Design object manipulation.** The following approaches are briefly discussed: case-based reasoning, machine learning, model based design, and atomic modifications.

- *Case-based reasoning*. One approach used in design processes is case-based reasoning (CBR). For an overview of theories, formalisations, techniques and applications, see (Kolodner, 1993; Hunt and Miles, 1994; Watson and Marir, 1994; Marir and Watson, 1994); for an overview of design applications that employ case-based reasoning, see (Maher and de Silva Garza, 1997) and for an overview of reuse in CBR see (Voß, 1997). Case-based reasoning is viewed as a re-design process for the 'adaptation' of a case. CBR is an integral part of the process of (re-)design (e.g., Daube and Hayes-Roth, 1989): previous design object descriptions are inspected and a promising design object description is modified to achieve requirements. A survey of design applications (Maher and de Silva Garza, 1997) based on CBR shows that a partonomic hierarchy is often employed to structure cases; a fact also noted in surveys on case retrieval (Altmeyer and Schürmann, 1996; Gebhardt, 1997).
  Case-based reasoning offers strategies for searching through histories of past cases (e.g., Dearden and Harrison, 1993; Gebhardt, 1997) such as similarity assessment and classification algorithms, and strategies for the adaptation of cases (e.g., Mostow, 1989; Carbonell, 1983; Voß and Oxman, 1996). Case-based reasoning has been applied to the domain of component-based systems (e.g., Rist, 1995; Takahashi, Oono, Saitoh and Matsumoto, 1995; Maher and de Silva Garza, 1997), which is, however, mostly concerned with the manipulation of design object descriptions.
- *Machine learning*. Machine learning is another technique applied in design; it provides a means to use previous design solutions. These solutions contain explicit and implicit knowledge that designers can interpret in new situations according to their own needs (Duffy, 1997). Some approaches learn control knowledge for systems (e.g., Minton, 1990; Straatman 1997). Other approaches attempt to learn strategic knowledge from previous decisions (e.g., Reich, 1993; Reich and Fenves, 1995). In yet another approach problem solving concepts are learned by reflecting on problem solving (Stroulia and Goel, 1994*b*).
- *Model-based design*. The compositional modelling approach described by Falkenhainer and Forbus (1991; 1992) is an approach to construct a model of an artefact on the basis of a description of the artefact and a query on the artefact. Queries are not further manipulated, but strategies are employed for re-construction of models. Extensions have been proposed, e.g., (Nayak, Joskowicz, 1996) within the manipulation of design parts of models. Although it is not considered to be a design or re-design task, compositional modelling can be viewed from that perspective. Strategies advocated by Falkenhainer and Forbus can be found e.g., in the construction of qualitative and quantitative simulation models (Sloof, 1998; Pos and Akkermans, 1996). A related approach is to transform a model into another model, e.g. by Generalised Directive Models (GDMs) (Heijst,

Terpstra, Wielinga and Shadbolt, 1992). GDMs are represented as a context sensitive rewrite grammar with which a model for problem solving can be refined.

- *Atomic modifications*. Another approach for design is proposed in (Gil and Tallis, 1995). It involves transaction-based manipulation of a design object description, with the notions of atomic transaction and composed transaction, and knowledge on the applicability of these transactions.

**Requirement manipulation.** Knowledge on the manipulation of requirements, expressed in terms of properties of artefacts, is part of the manipulation of sets of qualified requirements. Within some approaches this is termed 'functional reasoning': the function of an artefact can be described and reasoned about; for an overview of functional reasoning in design, see (Umeda and Tomiyama, 1997), for examples of knowledge on requirement manipulation employed in this thesis see, for example, Chapter 5 and (Brazier, Langen, Treur and Wijngaards, 1996; Brazier, Jonker, Treur and Wijngaards, 1998*c*). Knowledge of techniques with which the consistency of requirements can be checked are also of importance for detecting conflicting, imprecise, qualifications and/or requirements, see e.g., (Heitmeyer, Jeffords and Labaw, 1996); as well as knowledge of techniques with which predictions on future behaviour can be made, this can be done on the basis of simulation, e.g., (Rasmussen and Barrett, 1995).

**Design rationale and history navigation.** Representing and reasoning about design rationale can be part of the design process. As put by Lee (1997), a design rationale is an important tool because it can include not only the reasons behind a design decision but also the justification, the other alternatives considered, the trade-offs evaluated, and the argumentation that led to the decision. A brief overview of approaches to design rationale is given in (Moran and Carroll, 1996). Explicit representation of design rationale in a design process requires design decisions with respect to questions such as: how to construct a design rationale, how to reason about it, how to use design rationale to explain design decisions, etc. (e.g., Kumar, 1994; Stutt and Motta, 1995; Garlan, Allen and Ockerbloom, 1995; Vanwelkenhuysen, 1995; Vanwelkenhuysen and Mizoguchi, 1995; Brazier, Langen and Treur, 1997*c*). Design rationale can support re-use of previous sets of qualified requirements, design object descriptions, and re-use or previous design process strategies; see (Peña-Mora and Vadhavkar, 1996) for the employment of design rationale within the re-use of software (i.e., design object descriptions). Design rationale can aid in structuring design histories, and provides a guide for history navigation.

Within design processes one or more records can be kept, for example, to keep track of, e.g., when which modification was made, or to store partial designs (i.e., case libraries). One approach to navigation is to use explicit knowledge on the models stored in a history component to support query formulation (Galsey, Schwabacher and Smith, 1996). The retrieval of information from case libraries is an important aspect addressed within case-based reasoning. The ability of query reformulation (Fischer, Henninger and Redmiles, 1991; Fischer and Stevens, 1991) can be employed in the design process, e.g., when previous queries do not yield any results, or too many results have been returned. This activity is part of navigation within the design process (Logan and Smithers, 1992).

### 3.1.3 Software design

Within the field of Software Design, three related disciplines and domains of application are of interest: the disciplines of Requirements Engineering and Software Engineering, design of simulation models of physical systems, design of algorithmic software, and design of knowledge-based systems.

**Requirements Engineering & Software Engineering.** The discipline of Requirements Engineering (e.g., Davis, 1993; Sommerville and Sawyer, 1997; Wieringa, 1996) focuses on (the acquisition and specification of) requirements on software systems. It is very much a social process; requirements are elicited often by interviewing and observing humans (Zaff, McNeese, and Snyder, 1993; Sommerville and Rodden, 1994; Sumi, 1997). The acquisition, structuring and negotiation of requirements is common practice: the engineering of requirements is in fact requirements manipulation (Shaw and Gaines, 1995).

Various handbooks have been written on the design and re-design of software (e.g., Jackson, 1975; Sage and Palmer, 1990; Booch, 1991; Biggerstaff, 1992; Vliet, 1993; Gamma, Helm, Johnson and Vlissides, 1994; Mazza, Fairclough, Melton, de Pablo, Scheffer and Stevens, 1994; Riel, 1996; Pressman, 1997) within which many models are distinguished to structure the design process (Wieringa, 1996; Pressman, 1997); to name but a few: flow charts, entity-relationship models, object-oriented models. Some of these models involve an alternation between manipulating requirements and manipulating the software.

The process of software design is termed 'software development process' within Software Engineering and the term 'design' is reserved for the manipulation of the design object, i.e., a representation of software (which is later translated into code).

**Design of simulation models of physical systems.** In the domain of automated modelling, physical phenomena and their processes are described by simulation models of varying granularity, e.g., mathematical models, qualitative models, component models, etc. (Murray and Sheppard, 1988; Häuslein and Page, 1991; Akkermans, Borst, Pos and Top, 1995; Pos and Akkermans, 1996; Sloof, 1998). The simulation model formulation task can be viewed as a design task (Gruber, 1993): some with user interaction (e.g., Murray and Sheppard, 1988) and some fully automated (Pos and Akkermans, 1996; Sloof, 1998). Most of these approaches only model the manipulation of design object descriptions, a few also model the manipulation of requirements (e.g., Pos and Akkermans, 1996) to a very limited extent.

**Design of algorithmic software.** Tools have been developed for the (semi-)automatic design of software. While some tools have been developed to automatically design software (e.g., genetic algorithms are designed (Beck and Parmee, 1997), as well as an image-processing system composed of small image processing parts (Elst, Harmelen, Schreiber and Thonnat, 1995; Steier, 1991)), other tools have been developed to support system developers design software (e.g., programmer's assistants (Teitelman, 1986*a*; 1986*b*; Rich and Shrobe, 1986; Waters, 1986)).

In the above domains of application of (re-)design, the notion of compositionality is employed to structure the design object description (e.g., programmer's assistants often structure a design on the basis of the components distinguished by the intended user). Specific knowledge in these approaches may not be applicable to the subject of this thesis, but their methods and techniques are, to some extent, relevant.

**Design of knowledge-based systems.** In the domain of design of knowledge-based systems, most approaches only model the manipulation of design object descriptions (e.g., Korf, 1980; Baalen, 1992; Pirlein and Studer, 1995). Only a few approaches explicitly model strategies and requirement manipulation in their design process, e.g., (Wang, Rao and Zhou, 1995). In many approaches libraries are used during design; e.g., a library of models for diagnosis can be found in (Benjamins, 1993).

Within the discipline of knowledge engineering a number of design methodologies for knowledge-intensive systems have been developed (CommonKADS, VITAL, MIKE, PROTÉGÉ-II, TASK, RDR, KIDS, KNACK, and DESIRE). Each of these approaches describes phases and models to be used when (re-)designing knowledge-intensive systems. The methodology of the first eight modelling approaches is briefly outlined. The DESIRE approach is described in (Brazier, Jonker and Treur, 1998).

The *CommonKADS* (Common Knowledge Acquisition Development System) approach is a methodology for knowledge-based system development (Wielinga, Schreiber, Breuker, 1992; Hoog, Martil, Wielinga, Taylor, Bright, Velde, 1994; Schreiber, Wielinga, Akkermans, Velde and Hoog, 1994). Knowledge-based development within CommonKADS is based on the construction of a number of separate models (organisation model, task model, agent model, expertise model, communication model, design model) that capture the desired features of the system and its environment. The CommonKADS life cycle approach distinguishes phases, activities and products relevant for a knowledge-based system project.

The *VITAL* (Shadbolt, Motta, Rouge, 1993) approach to structured knowledge-based system development includes a knowledge engineering and a project management methodology. Within the project management the life cycle of an application project is modelled, by specific

process products. The life cycle configuration provides a mapping between management phases and the components of the knowledge engineering methodology.

The *MIKE* (Model-based and Incremental Knowledge Engineering) approach for the development of knowledge-based systems integrates semi-formal specification techniques, formal specification techniques (Fensel, 1995; Angele, Decker, Perkuhn and Studer, 1996), and prototyping into a coherent framework (Angele, Fensel and Studer, 1996). A life cycle approach is taken with the following phases: knowledge acquisition, design, implementation, and evaluation. The phase of design has the sub-phases requirement analysis, model construction and model evaluation. The phase knowledge acquisition has the sub-phases knowledge elicitation, interpretation and structuring, and formalisation. Two formal specification languages, P-KARL and L-KARL, are used to specify the reasoning process in detail.

The *TASK* modelling approach is designed to support the development of knowledge-based systems from conceptual specification to operationalisation (Pierret-Golbreich, 1993, 1994; Talon and Pierret-Golbreich, 1996*b*, Pierret-Golbreich and Talon, 1997). The TASK methodology includes task model oriented modelling, task centered representation (computational architecture), and a knowledge oriented acquisition method.

The *PROTÉGÉ-II* environment is a knowledge-acquisition shell that supports the construction of problem-solving methods using mechanisms as building blocks, modelling application tasks in terms of the constructed methods, generation of knowledge editors based on those task models, and the acquisition of knowledge from such knowledge editors (Musen, 1990; Puerta, Egar, Tu and Musen, 1992; Gennari, Altman, Musen, 1994, Eriksson, Puerta, Gennari, Rothenfluh, Tu and Musen, 1995). Within PROTÉGÉ-II approaches exist on how to build domain ontologies, domain independent methods and mapping relations. Generic building blocks exist for all three categories and reuse is an integral part of their methodology.

Ripple-down rules *RDR* is an approach to building and maintaining knowledge-based systems. The approach is based on test-analysis eliminating the need for knowledge engineering expertise during knowledge acquisition. RDR has been applied in domains of single- and multiple-classification tasks (Kang, Compton and Preston, 1998) and configuration / parametric design (Compton, Ramadan, Preston, Le-Gia, Chellen, Mulholland, Hibbert, Haddad and Kang, 1998). An underlying, fixed, problem solving method is the heart of RDR, which is applied in situations where test-cases are available.

The KIDS modelling approach includes a knowledge based software development system (Smith, 1990; 1991) in which a problem is defined by means of functional constraints on the input and output behaviour. A specification is created, which is refined by means of high-level transformations (selected by the user) which results in a more detailed (or optimised) specification. The focus lies on algorithms, and manipulation of a design object description; not the manipulation of requirements. Strategies are employed ('design tactics') which are used to optimise a specification.

The KNACK modelling approach (Klinker, Genetet and McDermott, 1990) includes a tool to create knowledge-based systems. This tool includes some manipulation of requirements: different perspectives of experts on a particular KBS can be manipulated according to some strategies.

## 3.2 Structuring principles for knowledge-intensive software systems

Structuring principles are not only employed to structure the design process, but also to structure the description (or actual configuration) of an artefact. The principle of 'hiding' parts of the structure at a lower abstraction level facilitates the manipulation of an artefact description: each level of the artefact description abstracts from lower level structures. The DESIRE approach to structuring principles is presented in Chapter 4.

One approach to artefact structures encountered in engineering disciplines, is system theory (proposed in the 1940's by the biologist Von Bertalanffy (1968) and its derivatives, often used to describe the structure of physical systems. Within this approach (hierarchical) components are distinguished, as well as interfaces with which components can be connected.

Ontologies employed to describe physical systems use this approach extensively, see, for example, (Alberts, 1993; Borst, Akkermans and Top, 1997).

Components are also used to describe the structure of more abstract artefacts, e.g., software systems. One approach to describing software is to describe the processes (e.g., JSD: Jackson, 1975). Algorithmic decomposition can be employed for this means. In this approach the task a process performs (i.e., its 'functionality') is composed of smaller tasks. Another approach often employed to describe software is object-oriented design. The components (the objects) model the *data*, and the interfaces of components (the methods of the objects) correspond to *processes* which modify the data. Algorithmic decomposition provides an ordering of the events within a system, object-oriented modelling emphasises relationships on the data without an explicit ordering of the events within a system; the two alternatives are orthogonal (Booch, 1991; Wieringa, 1996; Pressman, 1997).

Applications in knowledge-intensive domains are, for example, knowledge-based systems and multi-agent systems. The processes (e.g., tasks) performed, descriptions of sequencing of processes, descriptions of the information within the system, and knowledge employed to perform a task are often explicitly modelled within these systems.

Within the *CommonKADS* modelling framework (Wielinga, Schreiber, Breuker, 1992; Hoog, Martil, Wielinga, Taylor, Bright, Velde, 1994; Schreiber, Wielinga, Akkermans, Velde and Hoog, 1994) processes are explicitly represented, as is knowledge on the control of processes. Control knowledge resides at a distinct level: control knowledge over sub-processes is not private to a process. Knowledge and information descriptions are composed and explicitly related to processes. A (non-executable) formal representation language is available, with formal semantics based on dynamic logic. Partial operationalisation can be realised semi-automatically. An example of a CommonKADS description can be found in (Schreiber and Terpstra, 1996) in which an elevator configuration task is modelled. Other modelling frameworks based on the KADS-I 'philosophy' are the *VITAL* modelling framework (Shadbolt, Motta, Rouge, 1993) and the *MIKE* modelling framework (Fensel, 1995; Angele, Decker, Perkuhn and Studer, 1996; Angele, Fensel and Studer, 1996). An example of an elevator design task model modelled in VITAL can be found in Motta, Stutt, Zdrahal, O'Hara, and Shadbolt (1996). An example of an elevator design task model modelled in MIKE can be found in Poeck, Fensel, Landes and Angele (1996).

Within the TASK modelling framework (Pierret-Golbreich, 1993, 1994; Talon and Pierret-Golbreich, 1996*b*, Pierret-Golbreich and Talon, 1997), control knowledge is hidden at each level of process abstraction. The formal language with formal semantics in TASK cannot automatically generate prototype systems. An example of partial specifications of the VT task can be found in Talon and Pierret-Golbreich (1996*a*).

A modelling framework to describe software architectures (which is not a knowledge engineering modelling framework) is the *WRIGHT* specification language (Allen and Garlan, 1996). The specification language is developed for the formal specification of architectural styles in software (Garlan and Shaw, 1994) with which abstract behaviour of architectures can be described. Components, connectors and configurations are distinguished. Process hiding is achieved within the WRIGHT specification language; knowledge-hiding is not. If an architecture is sufficiently constrained an executable system may be generated (semi-)automatically.

For a more elaborate comparison of, among others, structuring principles for modelling frameworks for knowledge-intensive systems with explicit representation of knowledge, see Brazier and Wijngaards (1997, an extended version in (Brazier and Wijngaards, 1998)).

## 3.3 Example application domains

Two application domains are used in this thesis to illustrate the re-design of compositional systems (see Chapters 11 and 12): a knowledge-based system for diagnostic reasoning, and a multi-agent system. In this chapter a brief characterisation is given of each of these domains of application.

### 3.3.1 Knowledge-based system for diagnostic reasoning

Most systems for diagnostic tasks described in the literature adhere to the following static definition of diagnosis:

> Suppose one is to give a description of a system, *together with an observation of the system's behaviour* which conflicts with the way the system is meant to behave. The diagnostic problem is to determine those components of the system which, when assumed to be functioning abnormally, will explain the discrepancy between the observed and correct system behaviour. (Reiter, 1987). (italics added when including the quote in this section)

Or, in short, the expected behaviour of an artefact differs from the actual ('real') behaviour of the artefact and the diagnosis is the explanation of the difference in behaviour. Many formalisations of diagnosis have been proposed, e.g., (Reiter, 1987; Console and Torasso, 1990, Teije and Harmelen, 1994; Lucas, 1996) and many diagnostic systems have been designed, see (Benjamins, 1993) for a categorisation. All of these approaches take the observations of faulty and correct behaviour for granted and do not cover the decision making process to acquire observations during the diagnostic process nor is the relevant strategic reasoning explicitly incorporated in a process of diagnostic reasoning.

The example in this thesis focuses on a diagnostic process model, including explicit decision making on observations during the diagnostic reasoning: a diagnostic process is initiated on the basis of a complaint (i.e., observed abnormal behaviour); if additional information is needed to determine a diagnosis, specific observations are made by the system (e.g., by questioning the user of the system, or by directing and reading sensors).

One way to model a diagnostic process is by hypothetical reasoning. A possible diagnosis is assumed, and consequences are determined. These consequences are verified by observations on the actual behaviour of the artefact and the possible diagnosis is evaluated. A (partial) view of processes involved in a diagnostic reasoning system (Brazier, Langen, Treur and Wijngaards, 1996) is shown in Figure 3.2, based on a model of diagnostic reasoning as described in (Brazier, Treur and Wijngaards, 1996*a*; for an extended version see: Brazier, Jonker, Treur and Wijngaards, 1999*b*) and a logical description in (Treur, 1993).

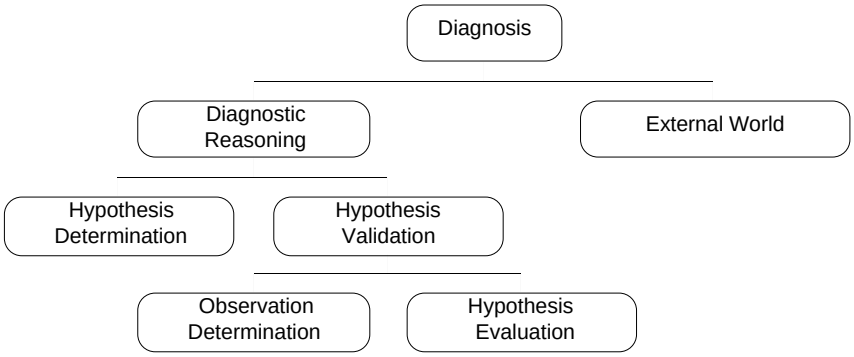


**Figure 3.2** Partial view of processes in a diagnostic system.

The processes shown in Figure 3.2 are briefly characterised below:

- Diagnostic Reasoning and External World are part of the diagnostic process: the Diagnostic Reasoning process generates focussed observations and (finally) determines diagnoses. The External World process performs observations and provides results of the observations requested.
- The Diagnostic Reasoning process is composed of two processes: Hypothesis Determination generates possible hypotheses (the hypotheses on which to focus), and Hypothesis Validation validates these hypotheses, by initiating further observations.
- The Hypothesis Validation process is composed of two processes: Observation Determination determines which observations related to the current hypotheses in focus are to be initiated, and Hypothesis Evaluation assesses these hypotheses on the basis of observation results.

An advantage of diagnostic reasoning as an application domain is that diagnostic reasoning has a relatively long research tradition in Artificial Intelligence and is therefore relatively well-analysed.

### 3.3.2 Self-modifying multi-agent system

Distributed processes are manifold in the real world. The (multi-) agent paradigm provides a means to characterise autonomous distributed processes. The systems (either human or automated) responsible for these processes are the *agents* in the multi-agent system. Each agent has its own environment, consisting of other agents and a material world. Agents are able to communicate with each other, can co-operate to jointly perform tasks, interact with the world (observe and/or act), and perform specific tasks. Some agents interact directly with humans, other agents interact with automated agents only (Kautz, Selman and Coen, 1994). In the near future the co-operation among agents and humans is expected to have impact on social conventions in society (Norman, 1994).

During the past years extensive research has been conducted in the field of multi-agent systems. Different notions of agency have been proposed (e.g., Nwana, 1996; Wooldridge and Jennings, 1995; Shoham, 1993). One notion of agents in which weak agency is distinguished from strong agency has been proposed by Wooldridge and Jennings (1995): weak agency is characterised by autonomy, social ability, reactiveness, and pro-activeness. In contrast the notion of strong agency is based on the characteristics of mentalistic and intentional notions (related to the notion of intentional stance by Dennett, 1987).

The characteristics of weak agency defined by Wooldridge and Jennings (1995) provide a means to reflect on the tasks an agent needs to be able to perform. Pro-activeness and autonomy are related to an agent's ability to reason about its own processes, goals and plans. Reactivity and social ability are related to the ability to interact with the material world and to communicate with other agents. The ability to communicate and co-operate with other agents and to interact with the material world often relies on an agent's ability to acquire and maintain its own knowledge of the world and other agents.

Agents, and multi-agent systems, are currently widely studied. Recent publications on agents include literature on software agents, e.g., see (Bradshaw, 1997), and literature on agent technology, e.g., see (Jennings and Wooldridge, 1998). Information brokering and information gathering agents (Levy, Sagiv and Srivastava, 1994; Sycara and Zeng, 1996; Knoblock and Ambite, 1997; Jonker and Treur, 1998a), a special kind of agent, play an important role in exploiting agent technology in the context of the Internet. Information gathering agents are sometimes developed 'ad hoc', or can be developed in a structured manner.

The agent metaphor offers a means to model situations with distributive activity on a conceptual level. Multi-agent systems have been proposed to model collaborative tasks such as design (Edmonds, Candy, Jones and Soufi, 1994; Vanwelkenhuysen and Mizoguchi, 1995; Dunskus, Grecu, Brown and Berker, 1995; Berker and Brown, 1996), and computer-based training systems (Boy, 1995).

The agent metaphor can also be used to develop agents that are able to dynamically design and create new agents, or to dynamically modify existing agents. For example, Internet agents that are capable of dynamically creating new agents to assist them in information gathering, or agents that are capable of creating interface agents tuned to specific users, are agents of this type. Also agents (including users) may be given the ability to influence the agent which re-designs the multi-agent system: requirements, partial design object descriptions and process objectives can be communicated and negotiated. As an example, consider humans explaining to their personal assistant which strategies to employ when processing e-mail on their behalf (Terveen and Murray, 1996).

Literature which partially addresses the topics 're-design of compositional systems' and 'self-modification' includes approaches based on genetic programming and parametric design, approaches based on meta-level architectures, and approaches based on mind-matter interactions. These approaches are described below.

Approaches based on *genetic programming & parametric design*. Most of the research in the area of dynamic agent creation is based on a genetic programming approach; e.g., (Cetnarowicz, Kisiel-Dorohinicki, and Nawarecki, 1996; Numaoka, 1996): design descriptions of agents are combined to evolve to a most suitable design description of an agent, according to

some criteria. Modifying problem solving methods by means of parametric design is an approach taken by (Teije, Harmelen, Schreiber and Wielinga, 1996) in which parameters of an otherwise fixed problem solving method are given appropriate values. In the genetic programming & parametric design approach a modified system is acquired by changing parameters of the system according to the modifications in the design description.

Approaches based on *meta-level architectures*. A reflective approach, in which an agent reasons about its own representation and re-designs this representation, is taken by e.g., (Schubert, 1997; Stroulia and Goal, 1994*a*; 1994*b*). A model-based approach to self-configuration of autonomous (spacecraft) systems is taken by (Williams and Nayak, 1996). Adapting a fixed task structure for different situations has been described by (Stroulia and Goel, 1994*a*). Reflecting on a problem solving method has been described by (Harmelen, Wielinga, Bredeweg, Schreiber, Karbach, Reinders, Voß, Akkermans, Bartsch-Spörl, and Vinkhuyzen, 1992; Teije and Harmelen, 1996). Modification of control knowledge in a problem solving method on the basis of inspection of the performance of the control knowledge is described by (Straatman, 1997).

Approaches based on *mind-matter interactions*. Self-modification entails the re-design of an agent's own description on the basis of a relationship between the actual 'physical' description of oneself and the dynamic flow of information within one's thought processes (Jonker and Treur, 1997). The emphasis is that to create new agents, an existing agent must be capable of designing a new agent on the basis of a model for design and then be capable of bringing this agent to life by performing actions modifying the material world. The integration of re-design on a conceptual and logical level (the mind aspect), and run-time modification of the system at the implementation level by performing material actions (the matter aspect) is of importance.

The domain of self-modifying multi-agent systems is a rich domain of application for re-design. It provides a natural setting for a process of re-design: an existing multi-agent system is re-designed by one (or more) of its agents. In Chapter 12 a description is given of a process of self-modification of a multi-agent system, based on a design agent which is part of that multi-agent system.

## 3.4 Discussion

The literature described in this chapter is divided into literature on structuring principles for design processes and literature on structuring principles for (knowledge-intensive) software systems. In addition two example domains of application are described.

On the basis of the literature on structuring principles for design processes the following essential elements of design processes can be distinguished: representation of design objects, representation of (qualified) requirements, manipulation of descriptions of design objects, manipulation of sets of (qualified) requirements, co-ordination of design processes, and representation of strategies. Combining these essential elements into a single model for design has been realised by the generic design model adopted for this thesis (see (Brazier, Langen, Ruttkay and Treur, 1994) and Chapter 6). Other design models, discussed in this chapter, do not necessarily include all these essential elements.

Additional conclusions regarding literature on design are the following:

- Models for design processes mostly focus on the manipulation of design object descriptions. Some involve the manipulation of sets of (qualified) requirements, and only a few also address the co-ordination of a process of design. In Requirements Engineering and Software Engineering, however, the manipulation of both requirements and design object descriptions is often addressed.
- Several methods and techniques are described which are applied in design processes: design strategies, design object manipulation (case-based reasoning, machine learning, model-based design, atomic modifications), requirement manipulation, and design rationale. Approaches for these methods and techniques provide insight in concepts and knowledge that play a role in a design process. Unfortunately, these methods and techniques have not yet been integrated into one framework within which strategical

knowledge can be employed to use a type of knowledge currently most suitable to the problem at hand.
- In the design of simulation models for physical systems, design of knowledge-based systems, and design of software (algorithms) a structured description of the design object is employed. Specific knowledge in these approaches may not be applicable to the subject of this thesis, but their methods and techniques are relevant to some extent. Explicit manipulation of requirements and their qualifications is often not present in these models, and, if present, severely limited.
- Most Knowledge Engineering methodologies distinguish activities which involve re-use, generic components, and libraries of components.

The desiderata identified in Section 2.2, provide a means to compare several approaches to structuring principles for knowledge intensive systems. The modelling frameworks CommonKADS, VITAL, MIKE, TASK, and WRIGHT have been discussed and briefly compared. Only parts of the desiderata (formulated in Section 2.2) have been achieved by these modelling frameworks, as shown in Table 3.1.

Modelling frameworks described in Section 3.2 incorporate some of the types of knowledge and representation (distinguished in Section 2.2 and shown in Table 3.1) needed to describe a compositional structure. Table 3.1 contains an overview of those desiderata. The notations and their interpretations are: 'explicit' indicates explicit realisation of the desideratum, 'partial' indicates a partial realisation of the desideratum, and 'none' indicates no realisation at all of the desideratum.

| *No.* | *Desideratum* | COMMONKADS | VITAL | MIKE | TASK | WRIGHT |
|---|---|---|---|---|---|---|
| *ds1* | Unambiguous, formal, representation language. | explicit | partial | explicit | partial | partial |
| *ds2* | Formal semantics. | explicit | partial | explicit | partial | partial |
| *ds3* | Explicit representation of both static and dynamic properties. | partial | none | partial | none | partial |
| *ds4* | Automated operationalisation. | partial | partial | partial | partial | partial |
| *ds5* | Compositionality as structuring principle | partial | partial | partial | partial | partial |

**Table 3.1** Overview of realisations of desiderata per modelling framework.

A conclusion from Table 3.1 is that not one of the described modelling frameworks realises these desiderata. A modelling framework which does explicitly realise these desiderata is the DESIRE modelling framework. This framework is described in Chapter 4. The compositional development method DESIRE is used in this thesis for both a description of compositional systems and as design objects within a process of design (Chapters 8, 9, 10, 11 and 12), and the description of the model of this process of re-design as a compositional system (Chapters 7, 8 and 9).

Two example application domains have been described: diagnostic reasoning and self-modifying multi-agent systems. These application domains can be briefly characterised as follows. Diagnostic reasoning includes explicit decision making on the observations to be performed during the diagnostic reasoning. The self-modifying multi-agent system includes re-design of a multi-agent system. The two example application domains are used to illustrate the re-design process described in this thesis.

In sum, the structuring principles for design processes and structuring principles for design objects as discussed in this chapter do not address all the desiderata involving these two notions (see Section 2.1 and Section 2.2). Compositional structuring principles including compositionality of processes and compositionality of knowledge are described in Chapter 4. Properties of such compositional structures are addressed in Chapter 5. A (compositional) generic model of design (Chapter 6), including the manipulation of sets of requirements and the

co-ordination of a process of design, is specialised into a (compositional) model of re-design (Chapters 7, 8, and 9) which is employed for the re-design of compositional systems: a diagnostic reasoning system (Chapter 11) and an example multi-agent system (Chapters 10 and 12). In the next chapter the application domains used as examples in this thesis are described in more detail.

Part **II**

# Compositional Systems: Structure & Properties

In this part compositional systems are described: their structure and their properties. In Chapter 4 a representation formalism for the structure of a compositional system for knowledge-intensive systems is presented. In Chapter 5 properties of compositional systems for knowledge-intensive domains are proposed. Both chapters include examples of structure and properties for the two domains of application: diagnostic reasoning systems and multi-agent systems.

# 4 Compositional Systems: Structure

A structure for compositional systems is described in this chapter which effectuates the desiderata identified in Section 2.2: ds1, ds2, ds3, ds4, and ds5. The realisation of desideratum dr1 ("representation of a knowledge-intensive system as a design object description") specifically depends on the realisation of these desiderata.

First the distinction between compositionality of processes and compositionality of knowledge is addressed in Section 4.1. Then process compositionality is addressed in Section 4.2, and knowledge compositionality in Section 4.3. In Section 4.4 the relation between process composition and knowledge composition is discussed. Generic models expressed in the compositional structure described in this chapter are addressed in Section 4.5. The formal semantics underlying the structure of compositional systems is briefly described in Section 4.6, and in Section 4.7 this chapter is concluded.

The two domains of application described in Section 3.3 are used to illustrate compositional systems in this chapter. The first example compositional system is a knowledge-based diagnostic reasoning system with which novice users of a washing machine can detect a flaw in their use of the washing machine. The second example compositional system is a multi-agent system in which a personal assistant plays an important role. A personal assistant is an agent which communicates with one or more users, and communicates with other agents or interacts in the world, on behalf of its user(s).

Some of the material in this chapter has been previously published in:

- *Formal Specification of Hierarchically (De)Composed Tasks* (Brazier, Treur, Wijngaards and Willems, 1995); a declarative description of hierarchically (de)composed tasks including both processes and knowledge.
- *Temporal Semantics of Compositional Task Models and Problem Solving Methods* (Brazier, Treur, Wijngaards and Willems, 1999); a description of a compositional system's behaviour; a temporal approach provides a means to describe the dynamics involved.
- *Principles of Compositional Multi-Agent System Development* (Brazier, Jonker and Treur, 1998); principles of a development method for compositional multi-agent systems.

## 4.1 Compositionality of processes and knowledge

As discussed in Chapter 2, the distinction between compositionality of processes and compositionality of knowledge is important. Both processes and knowledge can be described by compositional structures.

A *process* is viewed as an activity (for example, with which a task is brought to an end). As each process can, in turn, be composed of other processes, *levels of process abstraction* can be distinguished. A process can be considered on its own, or as a combination of a number of internal processes. If a process is composed of a number of internal processes, then these internal processes reside at a lower level of abstraction. The internal (lower-level) processes are responsible for the realisation of the (outer, higher-level) process. A *composition relation* defines how the processes are combined. Compositionality of processes is discussed at more length in Section 4.2.

Similarly, *levels of knowledge abstraction* can be distinguished. A *composition relation* determines how (lower-level composition of) knowledge structures can be combined to acquire (higher-level) composition of knowledge structures. Ontologies, used to express knowledge, can be structured as composed of other ontologies. Compositionality of knowledge is discussed in Section 4.3.

## 4.2 Process composition

A process composition describes the relationships: the processes, the levels of (process) abstraction, and the compositionality of each process.

### 4.2.1 Identification of processes at different abstraction levels

Two perspectives can be employed for the identification of processes at different levels of abstraction: a task perspective and a multi-agent perspective.

- In the *task perspective* a task is described in terms of processes needed to perform the task.
- In the *multi-agent perspective* processes within and between agents, within the external world, and between agents and the external world, are distinguished.

The processes identified in the task perspective are delegated to agents and the external world (identified in the multi-agent perspective). Process composition describes a one-to-many relation between processes. It can be described by means of a table (not shown) or depicted as a tree structure (see Figure 4.3 and Figure 4.4) or a box-in-box structure (see Figure 4.7). Processes are represented by *components*.

**Specification of abstraction levels.** The identification of levels of abstraction of processes results in two kinds of components: components *composed* of other components and *primitive* components. A primitive component can be a reasoning component (e.g., based on a knowledge base) or an alternative specification (e.g., based on a calculation, an optimisation, …).

Below two examples are given of processes at different levels of abstraction.

**Example diagnostic system: processes and process abstraction levels**

For the diagnostic process model the following processes may be distinguished at different levels of process abstraction, as shown in Figure 4.1, based on (Brazier, Jonker, Treur and Wijngaards, 1999*b*) and (Treur, 1993). In the latter, diagnostic reasoning has two sub-processes hypothesis selection and test selection. Test evaluation is a separate process at the same level of abstraction as diagnostic reasoning. In the current model hypothesis selection is replaced by hypothesis determination, test selection is replaced by observation determination, and test evaluation is replaced by both external world and hypothesis evaluation.

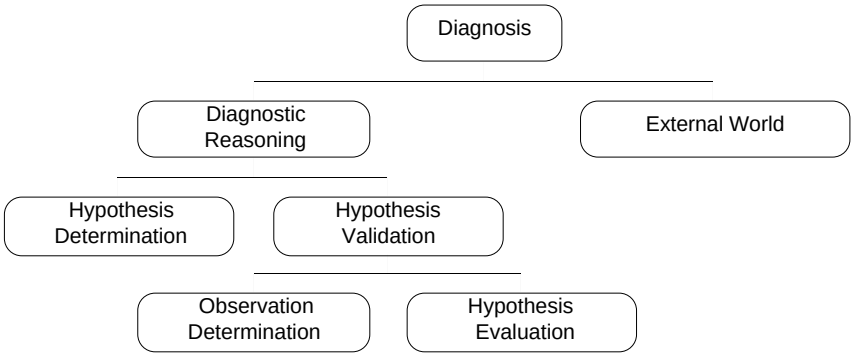


**Figure 4.1** Partial view of levels of process abstraction for diagnosis.

A diagnostic process includes not only the diagnostic reasoning process, but also the process of the acquisition of observation results within the external world.

The process of diagnosis involves the determination of one or more hypotheses on which to focus and the validation (i.e., confirming or rejecting) of these focus hypotheses on the basis of observations. The validation of hypotheses involves the determination of relevant (for the focus hypotheses) observations to be performed, and the evaluation of the results of the observations to validate the hypotheses in focus.

**Example multi-agent system: processes and process abstraction levels**

Within the multi-agent scenario three agents and an external world are distinguished, as shown in Figure 4.2. Within the personal assistant the same processes are distinguished as in the generic agent model in (Brazier, Jonker and Treur, 1996).

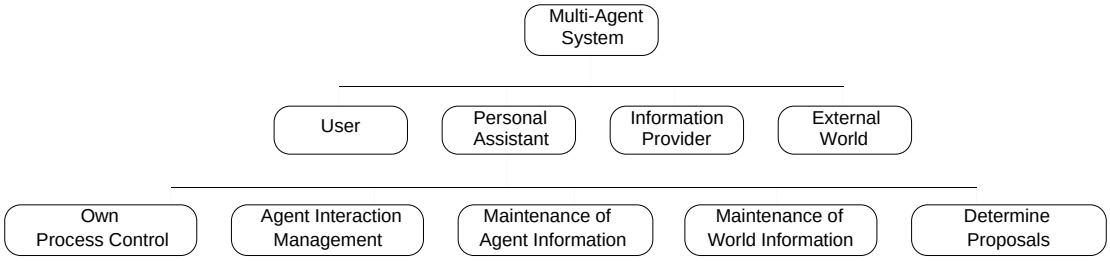


**Figure 4.2** Processes at different levels of abstraction in the example multi-agent system.

Three classes of agents are distinguished: users, personal assistants and information providers. In this example, one user is assumed to communicate with one personal assistant that can consult one information provider, or interact with the external world.

**Identification of a process.** Conceptually, for each process, the *types of information* required as *input* or generated as *output* of a process are specified in the *input and output interfaces* of a component. Names are defined for information types, and relations express which information is related to a component's output and/or input. In a pictorial representation, each component is annotated with the types of information in its input and output interfaces, for example, with input to the left and output to the right as shown in Figure 4.5 or in a table as shown in Figure 4.1.

Below examples of input and output information types for the processes distinguished for the diagnostic reasoning system (see Section 4.1) and the multi-agent system (see Section 4.2) examples are depicted in Figure 4.1 and Figure 4.2, respectively.

**Example diagnostic system: interface information types**

Input and output information types are distinguished within the process Diagnosis as shown in Table 4.1.

| *process* | *input information type* | *output information type* |
|---|---|---|
| Diagnostic Reasoning | • observation results | • assessed hypotheses<br>• required observations |
| External World | • required observations | • observation results |
| *Sub-processes of Diagnostic Reasoning* | | |
| Hypothesis Determination | • assessed hypotheses<br>• previously selected hypotheses<br>• interpreted observation results | • selected hypotheses |
| Hypothesis Validation | • focussed hypotheses<br>• observation results | • assessed hypotheses<br>• interpretations of observation results<br>• selected observations |
| *Sub-processes of Hypothesis Validation* | | |
| Observation Determination | • focussed hypotheses<br>• interpreted observation results | • selected observations<br>• predicted observation results |
| Hypothesis Evaluation | • focussed hypotheses<br>• observation results<br>• predicted observation results | • assessed hypotheses<br>• interpreted observation results |

**Table 4.1** Input and output information types of processes within the example diagnostic system.

The task Diagnostic Reasoning requires information on results of observations (observation results). The results of this task are assessments of hypotheses (assessed hypotheses) and observations to be performed (required observations).

The process External World accepts information on which observations are required (required observations) and provides information on results of observations (observation results).

The task Hypothesis Determination requires information on evaluations of hypotheses (assessed hypotheses), which hypotheses have been selected before (previously selected hypotheses) and interpretations of observations (interpretations of observation results). Information on selected hypotheses (selected hypotheses) is produced as a result of this task.

The task Hypothesis Validation requires information on which hypotheses to focus (focussed hypotheses) and results of observations that have been performed (observation results). During validation a need for specific information is identified (selected observations) depending, e.g., on the given focus hypothesis. When the validation process has terminated, the results of the process (hypotheses and observations that have been assessed) are available as output (i.e., assessed hypotheses and interpreted observation results).

The task Observation Determination needs information on which hypotheses to focus (focussed hypotheses) and the available information on observations (interpreted observation results). The results of this task are one or more observations to be performed (selected observations) and expected symptoms related to the hypotheses on which the task focussed (predicted observation results).

The task Hypothesis Evaluation needs information on observations performed (observation results), the hypotheses in focus (focussed hypotheses), and expected symptoms related to these hypotheses (predicted observation results). The results include an evaluation of the hypotheses (assessed hypotheses) and information on the observations performed (interpreted observation results).

**Example multi-agent system: interface information types**

In Table 4.2 input and output information types are distinguished for the agents and external world, and the internal processes within the personal assistant agent. The personal assistant agent is a *broker* of information, as depicted in Figure 4.8: information is collected from selected sources and distributed to interested parties, according to requests for and availability of information (Jonker and Treur, 1998*c*).

| *process* | *input information type* | *output information type* |
|---|---|---|
| user | • communication to user | • communication by user |
| personal assistant | • communication to PA<br>• observation results | • communication by PA<br>• selected observations |
| information provider | • communication to IP | • communication by IP |
| external world | • required observations | • observation results |
| *Within personal assistant:* | | |
| own process control | • belief info | • focus info |
| agent interaction management | • communication to PA<br>• belief info<br>• focus info | • communication by PA<br>• maintenance info |
| maintenance of agent information | • agent info | • agent info |
| world interaction management | • observation results<br>• belief info<br>• focus info | • selected observations<br>• maintenance info |
| maintenance of world information | • world info | • world info |
| determine proposal: agent specific task | • user interests<br>• product information | • proposal info |

**Table 4.2** Input and output information types of processes within the example multi-agent system.

The agents in this example are modelled on the basis of a generic model of an agent. Although it is not necessary to assume that all agents share the same ontology in their communication, for simplicity this assumption is made in this example.

The agent User accepts, as input, incoming communication from other agents (communication to user) and provides, as output, outgoing communication to other agents (communication by user).

The agent Personal Assistant accepts, as input, results of observations (observation results) and provides, as its output, selected observations to be performed (selected observations), all geared towards the External World. Furthermore, incoming communication from other agents is accepted as input (communication to PA) and outgoing communication to other agents is provided as output (communication by PA).

The agent Information Provider accepts, as input, incoming communication from other agents (communication to IP) and provides, as output, outgoing communication to other agents (communication by IP).

The External World receives, as input, observations to be performed (required observations) and produces results of these observations (observation results).

The component Own Process Control has, as input, information on beliefs (belief info) and generates a focus on, e.g., a user or scope of interests (focus info).

The component Agent Interaction Management has the same input as the agent itself (communication to PA) as well as information on beliefs and foci (belief info and focus info). The output produced includes the output of the agent as a whole (communication by PA) and information on other agents and the world that needs to be maintained within the agent (maintenance info).

The component Maintenance of Agent Information accepts, as input, information on other agents (agent info) and provides, as output, information on other agents (agent info).

The component World Interaction Management has, as input, results from observations (observation results) and information on beliefs and foci (belief info and focus info). As output, it produces observations which need to be performed (selected observations) as well as information on other agents and the world that needs to be maintained within the agent (maintenance info).

The component Maintenance of World Information accepts, as input, information on the world (world info) and provides, as output, information on the world (world info).

The component Determine Proposal: Agent Specific Task uses information on interests of users (user interests) and information on products (product information), and produces information on proposals to the users (proposal information).

In the input or output interface of a component, information types can be defined at different meta-levels. A separate part of the interface is reserved for each meta-level.

### 4.2.2 Process composition relation

The term ‘process composition’ refers to the relationship between a component and its sub-components. This relationship is functional in the sense that the behaviour of a composed component is specified by the composition relation and the behaviour of the sub-components (Brazier, Jonker, and Treur, 1998). The composition relation itself is discussed in this section. All components have the same, uniform, structure, as shown in Figure 4.3. A distinction is made between kernel information and task control information.

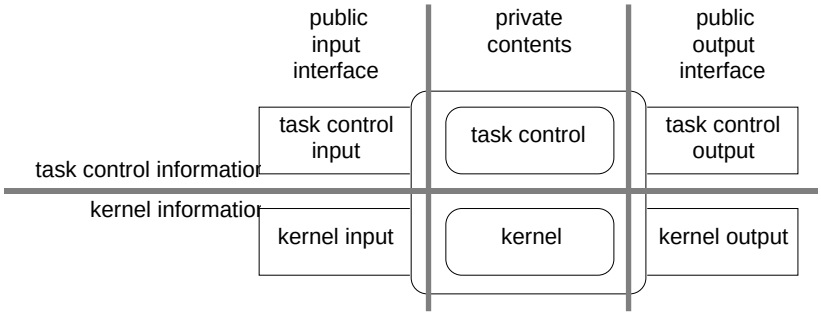


**Figure 4.3** Uniform structure of a component.

The input and output information types of a process are defined in the kernel input and output of a component. The kernel input and output interfaces are public. The internal contents of the component (other components or reasoning knowledge) is private. The distinction between

public and private is essential to information hiding. The kernel information of a primitive component can be specified by a knowledge base or an alternative specification. The kernel of a composed component is specified by components, information links and the task control of a composed component consists of knowledge about task control.

The task control information of a component specifies control within the component. The task control public input and output interface provide control information *to* (e.g., how to activate a component) and *from* a component (e.g., state is idle and the success of activation, i.e., if a given evaluation criterion has been reached). The private contents of the task control specify knowledge about activation of sub-components and information links in relation to success or failure of evaluation criteria.

Two views can be distinguished on composition of processes: a *static view* (i.e., information links), and a *dynamic view* (i.e., task control knowledge). The composition of processes is described by these two views

- information links between the component and its sub-components, and between the sub-components, and
- task control knowledge (of sub-components and information links).

**Information links.** Information links between components define the types of information transferred between components. More specifically, the relations expressing information links between components are explicitly specified and named. This abstracts from actual activation of information links: the control (dynamics) over the information links is defined in the task control of the encompassing component.

An information link is a directed channel for flow of information, possibly with a translation of *source information types* into *destination information types*. Two kinds of information links are distinguished, as shown in Figure 4.4.

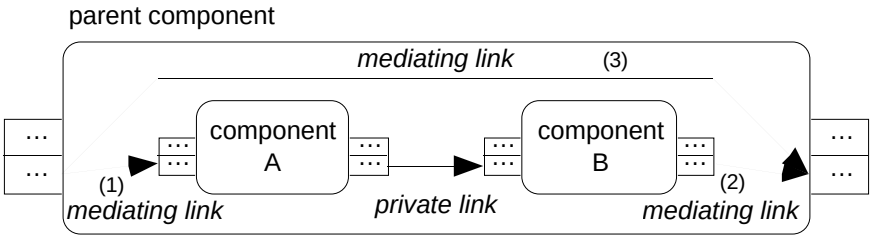


**Figure 4.4** Examples of private and mediating information links.

A *private link* defines an information link between two components linking the output interface of one component with the input interface of another component. A *mediating link* defines an information link between a parent component and a sub-component. In Figure 4.4, link (1) is a mediating link from the input interface of the parent component to the input interface of a sub-component, link (2) is a mediating link from the output interface of a sub-component to the output interface of the parent component, and link (3) is a mediating link within the parent component from its input interface to its output interface.

An example is given of information links within a process for each of the two example domains, as depicted in Figure 4.5, Figure 4.6 and Figure 4.7.

**Example diagnostic system: information links**

The information links in three levels of process abstraction are shown below for the example diagnostic system. First the information links in the component Diagnosis are depicted in Figure 4.5.

Within this component two private links are defined. Mediating links are not defined at this level of abstraction: the information in the component Diagnosis is self-contained: the component External World represents the external (observable) world for the component Diagnostic Reasoning.

- The private link observations to be performed transfers required observations from the output interface of Diagnostic Reasoning to the input interface of External World.
- The private link results from observations transfers observation results from the output interface of External World to the input interface of Diagnostic Reasoning.

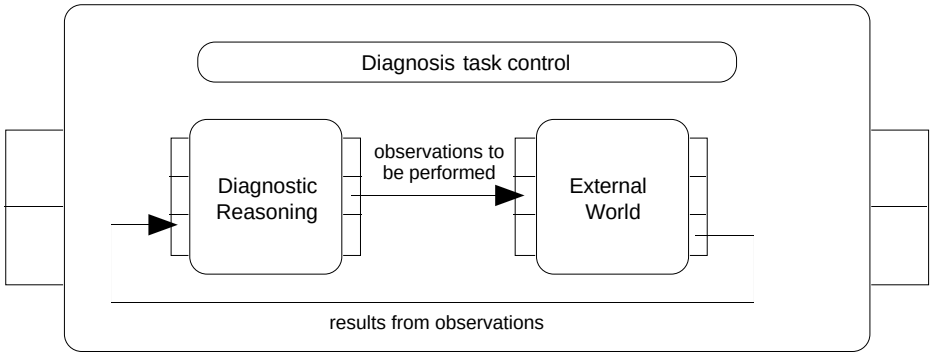


**Figure 4.5** Information links within the component Diagnosis.

The information links in the component Diagnostic Reasoning, a sub-component of Diagnosis, are shown in Figure 4.6.

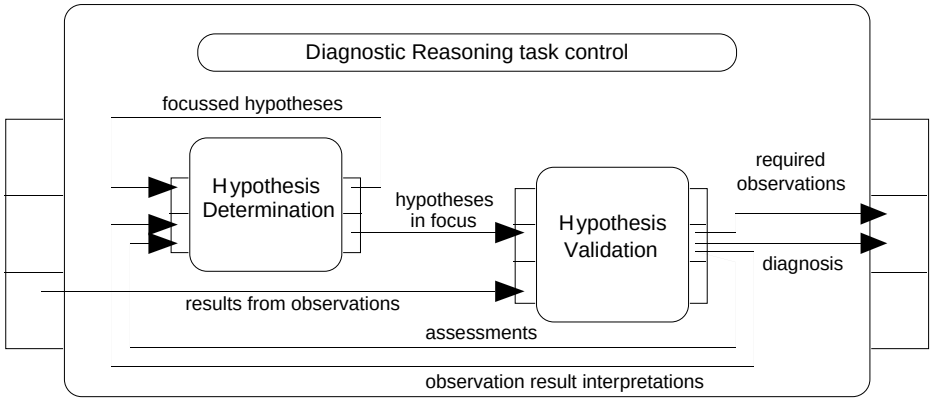


**Figure 4.6** Information links within the component Diagnostic Reasoning.

Within this component four private links and three mediating links are defined:

- The mediating link results from observations transfers observation results from the input interface of Diagnostic Reasoning to the input interface of Hypothesis Validation.
- The private link hypotheses in focus transfers selected hypotheses from the output interface of Hypothesis Determination to the input interface of Hypothesis Validation.
- The private link observation result interpretations transfers interpreted observation results from the output interface of Hypothesis Validation to the input interface of Hypothesis Determination.
- The private link assessments transfers assessed hypotheses from the output interface of Hypothesis Validation to the input interface of Hypothesis Determination.
- The private link focussed hypotheses transfers selected hypotheses from the output interface of Hypothesis Determination to previously selected hypotheses in the input interface of Hypothesis Determination on the basis of an explicit mapping between these information types.
- The mediating link diagnosis transfers assessed hypotheses from the output interface of Hypothesis Validation to the output interface of Diagnostic Reasoning.
- The mediating link required observations transfers selected observations from the output interface of Hypothesis Validation to required observations in the output interface of Diagnostic Reasoning on the basis of an explicit mapping between these information types.

The information links in the component Hypothesis Validation, a sub-component of Diagnostic Reasoning, are shown in Figure 4.7.

Within this component two private links and six mediating links are defined:

- The mediating link focus hyp to OD transfers focussed hypotheses from the input interface of Hypothesis Validation to the input interface of Observation Determination.

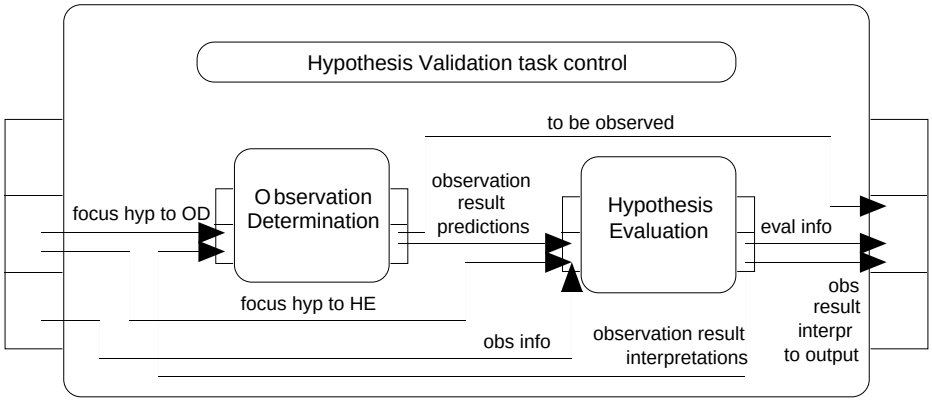


**Figure 4.7** Information links within the component Hypothesis Validation.

- The mediating link focus hyp to HE transfers focussed hypotheses from the input interface of Hypothesis Validation to the input interface of Hypothesis Evaluation.
- The mediating link obs info transfers observation results from the input interface of Hypothesis Validation to observation information in the input interface of Hypothesis Evaluation on the basis of an explicit mapping between these two information types.
- The mediating link to be observed transfers selected observations from the output interface of Observation Determination to the output interface of Hypothesis Validation.
- The private link observation result predictions transfers predicted observation results from the output interface of Observation Determination to the input interface of Hypothesis Evaluation.
- The private link observation result interpretations transfers interpreted observation results from the output interface of Hypothesis Evaluation to the input interface of Observation Determination.
- The mediating link eval info transfers assessed hypotheses from the output interface of Hypothesis Evaluation to the output interface of Hypothesis Validation.
- The mediating link obs result interpr to output transfers interpreted observation results from the output interface of Hypothesis Evaluation to the output interface of Hypothesis Validation.

Similarly, information links within the different levels of process abstraction can be shown for the example multi-agent system.

**Example multi-agent system: top-level composition**

The information links within the top-level of the multi-agent system are shown in Figure 4.8.

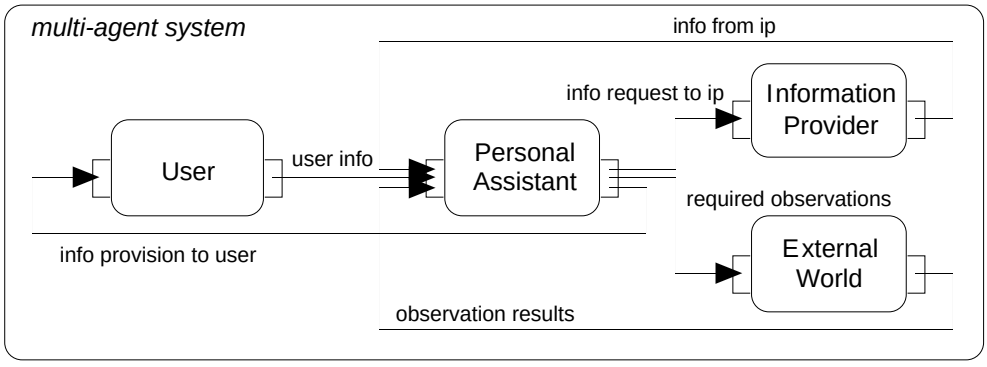


**Figure 4.8** Information links at the top-level of the multi-agent system.

Within this multi agent system six private links are defined, as shown in Figure 4.8. Mediating links are not defined at this level of abstraction: the processes distinguished in the multi-agent system do not need interaction with processes outside of the multi-agent system. The agent ‘personal assistant’ plays a central role in this multi-agent system; all communication is via this agent:

- The private link user info transfers communication by user from the output interface of User to communication to PA in the input interface of Personal Assistant, on the basis of an explicit mapping between these two information types.
- The private link info provision to user transfers communication by PA from the output interface of Personal Assistant to communication to user in the input interface of User, on the basis of an explicit mapping between these two information types.
- The private link info request to IP transfers communication by PA from the output interface of Personal Assistant to communication to IP in the input interface of Information Provider, on the basis of an explicit mapping between these two information types.
- The private link info from IP transfers communication by IP from the output interface of Information Provider to communication to PA in the input interface of Personal Assistant, on the basis of an explicit mapping between these two information types.
- The private link required observations transfers required observations from the output interface of Personal Assistant to the input interface of External World.
- The private link observation results transfers observation results from the output interface of External World to the input interface of Personal Assistant.

Information links within the Personal Assistant are based on the information links specified in the generic agent model in Section 4.5.2, Figure 4.17.

**Task control knowledge.** Information links within a composed component describe the static part of the composition relation; task control knowledge describes the dynamic part of the composition relation. Components and information links can be activated either sequentially or in parallel:

- *sequentially*: specific activation of components and information links, depending on temporal relations between components and information links;
- *continuously* (awake): whenever new input information is available, a component or information link becomes active.

Task control knowledge defines temporal relations between components and information links: which components must (directly) precede other components and which information link activation is required. Both components and information links have names, which are used to specify task control knowledge. Task control knowledge specifies under which conditions, which tasks and information links are activated. These conditions, preconditions for task activation, may, for example, include *evaluation criteria* expressed in terms of the evaluation of the results (success or failure) of one or more of the preceding tasks. General knowledge of task control is specified: knowledge of which tasks may be performed in parallel and which tasks must precede which other tasks (not necessarily directly nor conditionally). Task control knowledge is expressed in a temporal knowledge base, as shown in the examples below.

**Example diagnostic system: task control knowledge**

Two task control knowledge elements are shown below for the example diagnostic system. One example of task control knowledge resides within component Diagnostic Reasoning, the other example of task control knowledge resides within an internal component Hypothesis Validation.

A part of the task control knowledge of the component Diagnostic Reasoning:

```
if   previous_component_state( hypothesis_determination, active )
     and   component_state( hypothesis_determination, idle )
     and   evaluation( hypothesis_determination, hypos_determined, any, succeeded )
then      next_component_state( hypothesis_validation, active )
     and   next_link_state( hypotheses_in_focus, uptodate )
     and   next_link_state( focussed_hypotheses, uptodate );
```

This task control knowledge element describes a pre-condition for the activation of the component Hypothesis Validation: if component Hypothesis Determination has just become idle (i.e., previously it was active and currently it has become idle) and the component Hypothesis Determination has produced the specified results (i.e., a hypothesis is determined), then the component Hypothesis Validation is to be made

active in the next state, plus that two links are made up to date (i.e., have been activated) *before* the component Hypothesis Validation becomes active.

At a lower process abstraction level a task control knowledge element(part of the task control knowledge of the component Hypothesis Validation) is:

```
if      start
then    next_component_state( hypothesis_evaluation, awake )
  and   next_component_state( observation_determination, active )
  and   next_link_state( focus_hyp_to_OD, awake )
  and   next_link_state( obs_info, awake )
  and   next_link_state( focus_hyp_to_HE, awake );
```

The above example task control knowledge is activated the first time the component Hypothesis Validation *becomes* active (or awake), i.e., the 'start' of the component. This knowledge specifies that if the component is started, then in the next state the component Hypothesis Evaluation is to be made continually active (i.e., awake), and the component Observation Determination is to be made temporarily active (i.e., just active), plus three links have to be made awake. The result is that each information link transfers information whenever new information is available at the source of the information link.

**Example multi-agent system: task control knowledge**

In the example multi-agent system top-level task control knowledge is minimal. At the top-level of the multi-agent system one task control knowledge element can 'give life' to the processes within the multi-agent system:

```
if      start
then    next_component_state( user, awake )
  and   next_component_state( personal_assistant, awake )
  and   next_component_state( information_provider, awake )
  and   next_component_state( external_world, awake )
  and   next_link_state( user_info, awake )
  and   next_link_state( info_provision_to_user, awake )
  and   next_link_state( info_request_to_ip, awake )
  and   next_link_state( info_from_ip, awake )
  and   next_link_state( required_observations, awake )
  and   next_link_state( observation_results, awake );
```

In the above example task control knowledge all agents, the external world, and all information links are made continuously active (i.e., awake). The result is that each component is made awake and processes new information when it becomes available in its input interface. Each information link transfers information whenever new information is available at the source of the information link.

## 4.3 Knowledge composition

*Knowledge composition* is defined by knowledge structures at different levels of abstraction, and the composition relation between these knowledge structures. Note that process abstraction levels and knowledge abstraction levels are not tightly coupled: knowledge abstraction levels do not need to correspond to process abstraction levels.

In this section, first the identification of knowledge structures at different levels of abstraction is addressed, then the composition of knowledge structures.

### 4.3.1 Identification of knowledge structures at different abstraction levels

Two kinds of knowledge structures are distinguished: *information types* and *knowledge bases*. Both information types and knowledge bases can be identified for each level of abstraction level.

**Information types.** An information type defines (part of) an ontology (i.e., a lexicon, or vocabulary): objects, their sorts, and relations and functions on these sorts. The representation format can be graphical, e.g., based on conceptual graph-like structures (Jonker, Kremer, Leeuwen, Pan and Treur, 1998) or textual (concise). Both the graphical notation and the

concise, textual notation are used to depict a number of information types in the example domains.

**Example diagnostic system: information types**

Below parts of two information types are shown. Definitions of two relations for the information type domain hypotheses, and definitions of three relations for the information type domain observables are depicted.

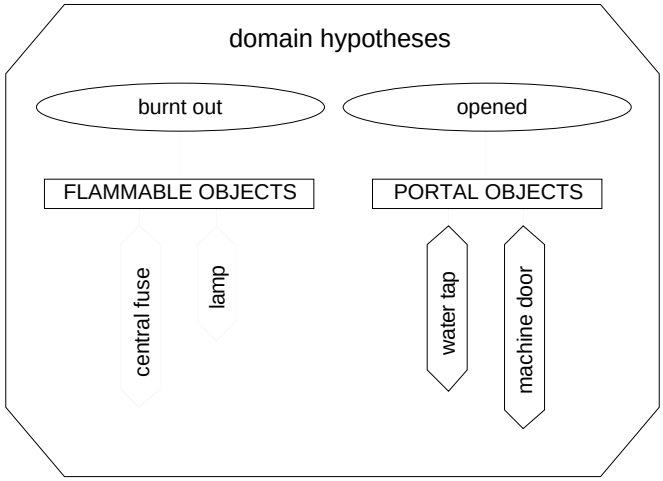


**Figure 4.9** Two relations defined in the information type domain hypotheses.

Two relations are defined in the information type in Figure 4.9. The relation burnt out has one argument of the sort FLAMMABLE OBJECTS. Two objects are defined in this sort: central fuse and lamp. The relation opened has one argument of the sort PORTAL OBJECTS. Two objects are defined for this sort: water tap and washing machine door.

The following statements can be formulated with these relations:

```
burnt_out( central_fuse);
opened( water_tap );
```

In Figure 4.10 three relations are defined in the information type domain observables.

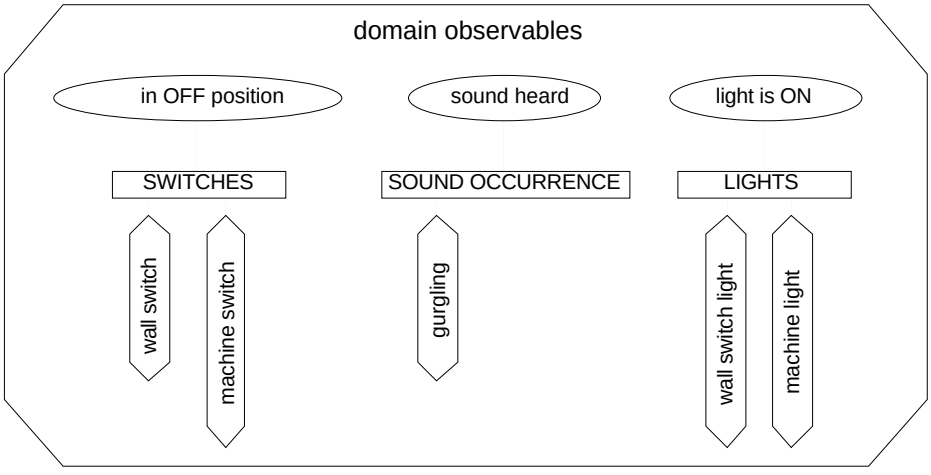


**Figure 4.10** Three relations defined in the information type domain observables.

The following statements can be formulated with these three relations:

```
in_OFF_position( wall_switch );
sound_heard( gurgling );
light_is_ON( wall_switch_light );
```

The information types depicted in Figure 4.9 and Figure 4.10 can also be represented in the textual notation.

**Example diagnostic system: information types in textual notation**

The information type domain hypotheses is specified as follows:

```
information type domain_hypotheses
    sorts
        FLAMMABLE_OBJECTS,
        PORTAL_OBJECTS;
    objects
        central_fuse,
        lamp: FLAMMABLE_OBJECTS;
        water_tap,
        machine_door: PORTAL_OBJECTS;
    relations
        burnt_out: FLAMMABLE_OBJECTS;
        opened: PORTAL_OBJECTS;
end information type
```

The information type domain observables is specified as follows:

```
information type domain_observables
    sorts
        SWITCHES,
        SOUND_OCCURRENCE,
        LIGHTS;
    objects
        wall_switch,
        machine_switch: SWITCHES;
        gurgling: SOUND_OCCURRENCE;
        wall_switch_light,
        machine_light: LIGHTS;
    relations
        in_OFF_position: SWITCHES;
        sound_heard: SOUND_OCCURRENCE;
        light_is_ON: LIGHTS;
end information type
```

Meta-levels of information can be distinguished. A single meta-level relationship can be specified by means of a *meta-description*. A meta-description, which extends the contents of a sort, offers a means to name a different information type, so that relations (and all their arguments) can be used as objects in this sort, thereby providing an explicit naming relationship.

**Example diagnostic system: meta-description**

In Figure 4.11 a meta-description is shown in the information type meta domain hypotheses information. In this information type, the contents of the information type domain hypotheses information become objects of the sort HYPOTHESES. The textual notation for this meta-description is shown below.

```
information type meta_domain_hypotheses_information
    sorts
        HYPOTHESES;
    meta-descriptions
        domain_hypotheses_information: HYPOTHESES;
end information type
```

Given these definitions relations can be defined using the sort HYPOTHESES, so that expressions can be formulated *about* domain hypotheses.

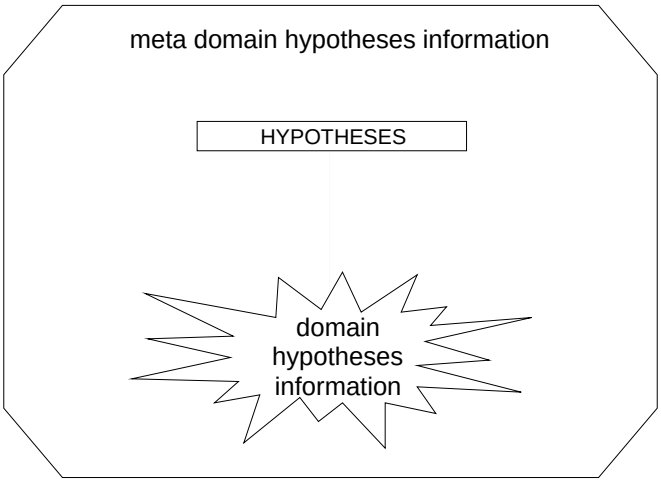


**Figure 4.11** Meta-description in the information type meta domain hypotheses information.

**Knowledge bases.** Relations between information types and knowledge bases are specified to indicate the ontology used within a knowledge base. A knowledge base defines the knowledge used in one or more processes.

Examples of knowledge bases are given for each of the examples used in this chapter.

**Example diagnostic system: knowledge bases**

The components Hypothesis Evaluation and Observation Determination both contain knowledge with which a relation is described between hypotheses and observations. In this example, the knowledge base of the component Hypothesis Evaluation contains *generic* knowledge, i.e., knowledge applicable in many situations (different domains). The component Observation Determination contains domain *specific* knowledge, i.e., knowledge geared towards a specific domain. Example instances of knowledge are shown below for both knowledge bases.

An example from the generic knowledge base hypothesis evaluation kb of the component Hypothesis Evaluation is the following:

**knowledge base** hypothesis_evaluation_kb
**if** focus_hypothesis( H: HYPOTHESIS )
  **and** hypothesis_is_related_to_observation( H: HYPOTHESIS, X: OBSERVATION )
  **and** predicted( X: OBSERVATION, S: Sign )
  **and** observation_has_been_performed( X: OBSERVATION, S: SIGN )
  **and** observed( X: OBSERVATION, Sobs: SIGN )
  **and** S: SIGN ≠ Sobs: SIGN
**then** rejected( H: HYPOTHESIS );
**end knowledge base**

This knowledge element expresses the knowledge that if, given a focus hypothesis, the predicted observation has been contradicted by observation, then the hypothesis is rejected, i.e., it is concluded that the hypothesis will not provide an answer to the given diagnostic problem.

Below two instances are given from the domain specific knowledge base observation determination washing machine kb of the component Observation Determination. The knowledge elements describe a causal relationship between hypotheses and symptoms.

**knowledge base** observation_determination_washing_machine_kb
**if** burnt_out( central_fuse )
**then** in_ON_position( wall_switch )
  **and** **not** light_is_ON( wall_switch_light );

**if** **not** opened( water_tap )
**then** **not** sound_heard( gurgling );
**end knowledge base**

The first knowledge element represents the knowledge that if the hypothesis is that the central fuse is burnt out then two symptoms must be present: the switch on the wall, connected to the washing machine, should be in the ‘on’ position, and the light in that switch on the wall should not be shining.

The second knowledge element represents the knowledge that if the hypothesis is that the water tap is not opened, then the symptom must be that there is no gurgling sound, of water rushing through the water-tube from the tap to the washing machine.

### 4.3.2 Composition relation for knowledge structures

Similar to the way components can be composed in a process composition, information types can be composed of other information types, and knowledge bases can be composed of other knowledge bases.

The composition of knowledge is shown in the example below for the composition of information types and for the composition of a knowledge base and information types.

**Example diagnostic system: composition of knowledge structures**

First an example is given of the composition of information types. Figure 4.12 shows relations between several information types: an information type on the left *refers to* an information type on the right when a connecting line is present. An information type can contain a meta-description of another information type, residing at a lower meta-level, as indicated by a dashed line.

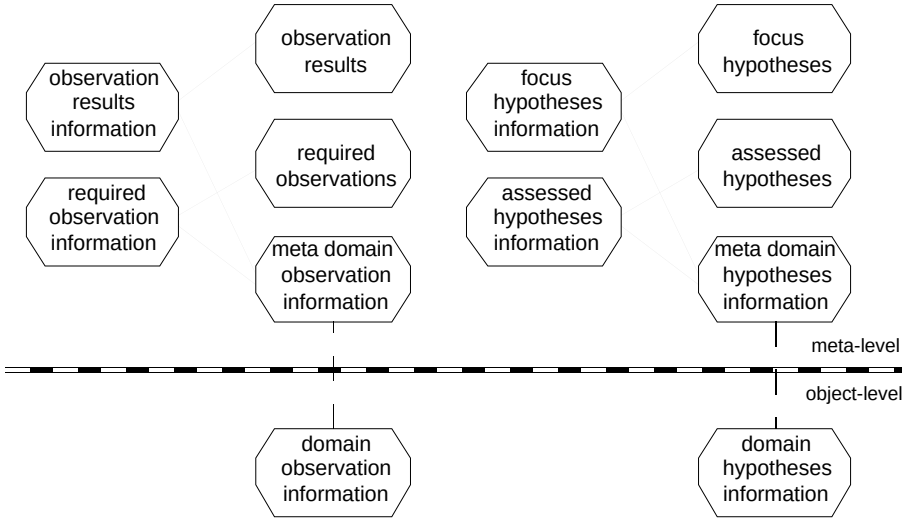


**Figure 4.12** Graphical representation of some of the information types in the example diagnostic system.

The information types domain observations and domain hypotheses both reside at the object-level. The information types meta domain observations and meta domain hypotheses contain meta-descriptions of these object level information types in the sorts OBSERVATIONS and HYPOTHESES, respectively. At the meta-level it is possible to formulate statements *about* information in the domain observations and domain hypotheses, e.g.,:

required_observation( in_OFF_position( wall_switch ), pos )

or

focus_hypothesis( burnt_out( central_fuse ), neg ).

The information type required observations contains definitions of generic relationships, in this particular case, the relation required_observation is defined on OBSERVATIONS. The information type required observation Information refers to two information types: required observations and meta domain observation Information, with as a result the extension of the sort OBSERVATIONS within the information type required observation information, so that the sort OBSERVATIONS used to define the relation required_observation contains the meta-description of the object level information type domain observation information. This definition of extending sorts gives a means to define a generic information type, and merge its contents with the contents of another, specific, information type providing a clear separation of generic vs. specific information while making explicit where the merged information resides.

The information types observation results, focus hypotheses, and assessed hypotheses all contain generic information. The information types observation results information, focus hypotheses information, and assessed hypotheses information extend generic information by defining a merger of generic sorts with sorts with (domain) specific content, available in the information types meta domain observation information and meta domain hypotheses information.

In Figure 4.13 an example is given of the composition of knowledge bases and information types.

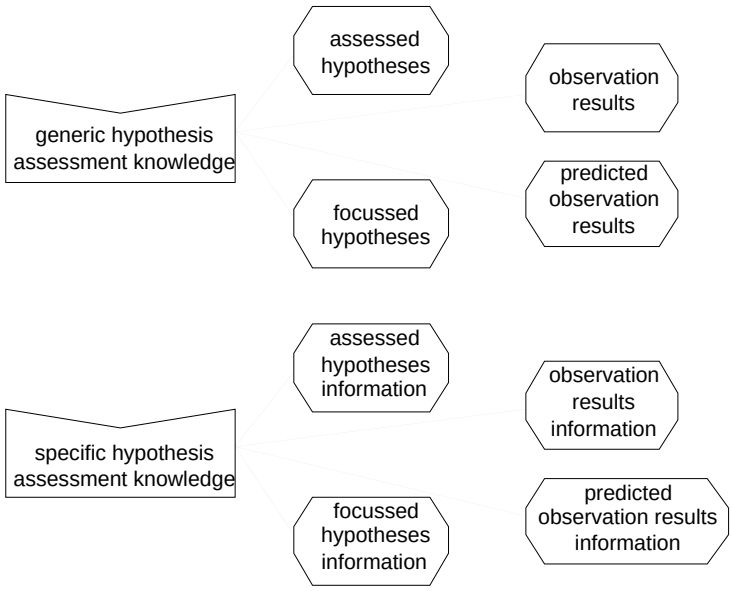


**Figure 4.13** Graphical representation of two instances of knowledge for hypothesis assessment: a knowledge base generic hypothesis assessment knowledge and a knowledge base specific hypothesis assessment knowledge, both referring to four information types (to be read from the left to the right).

Both the knowledge base generic hypothesis assessment knowledge and the knowledge-base specific hypothesis assessment knowledge make use of four information types, as shown in Figure 4.13. Note that there is no notion of input, output or internal information types; such indications are only valid when a knowledge base is viewed as a *process*.

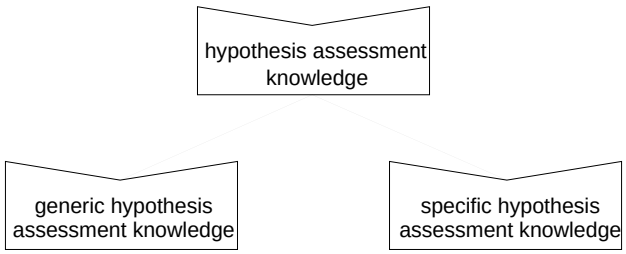


**Figure 4.14** Composition relation of three knowledge bases: the knowledge base hypothesis assessment knowledge refers to the knowledge bases generic hypothesis assessment knowledge and specific hypothesis assessment knowledge.

The relation between levels of knowledge abstraction is shown in Figure 4.14, in which the composition relation between three knowledge-bases is depicted (to be read from top to bottom). The knowledge base hypothesis assessment knowledge contains the combined knowledge of the knowledge bases generic hypothesis assessment knowledge and specific hypothesis assessment knowledge.

## 4.4 Relation between process composition and knowledge composition

Processes employ knowledge (information types in input and output interfaces, knowledge bases). Which particular knowledge structure is employed in which particular component is defined by the relation between process composition and knowledge composition (i.e., the cells in the matrix of Figure 4.15). Figure 4.15 shows a view on compositionality of processes and compositionality of knowledge as separate dimensions.

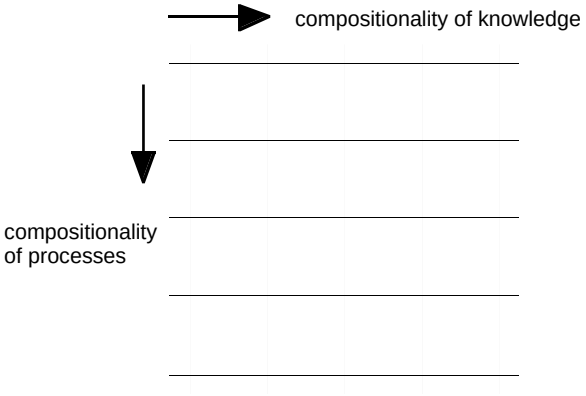


**Figure 4.15** Compositionality of processes and compositionality of knowledge

In the example below a process within the process composition is related to knowledge structures in the knowledge composition.

**Knowledge-based diagnostic reasoning system scenario**

In Figure 4.16 the process Hypothesis Evaluation makes use of several knowledge structures.

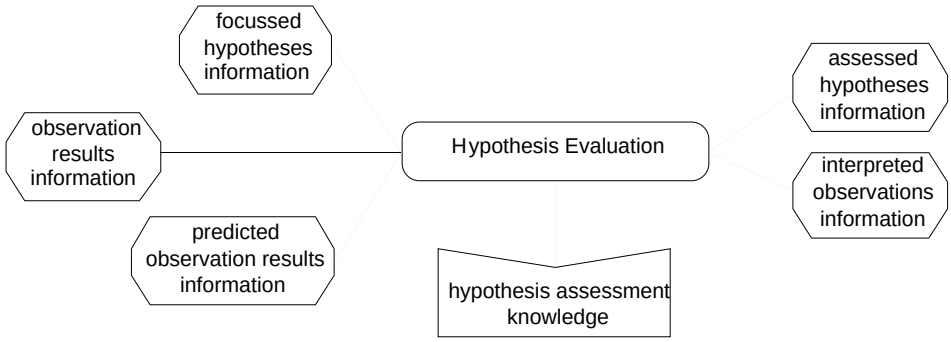


**Figure 4.16** The process Hypothesis Evaluation refers to input information types (on its left) and output information types (on its right) as well as a knowledge base (below the process).

The information types focussed hypotheses information, observation results information, and predicted observation results information are input information types for the process of hypothesis evaluation. The information types assessed hypotheses information and observation results information are output information types of that process, and the knowledge base hypothesis assessment knowledge is the knowledge base of that process.

## 4.5 Generic models

Reuse plays an important role during design and re-design of systems. Some parts of the structure of an object may be reused across applications. These generic parts may be defined by generic models.

Many approaches to the design of systems (and in particular software systems) distinguish generic architectures or models (Chandrasekaran, 1986; Kowalczyk and Treur, 1990; Breuker, 1994), architectural styles (Garlan, Allen and Ockerbloom, 1994; Garlan and Shaw, 1994), or design patterns (Gamma, Helm, Johnson and Vlissides, 1994; Riel, 1996). Most approaches define methodologies with which generic models are defined and/or (re-)used to build a domain dependent model for a specific domain of application.

The methodology incorporated in DESIRE, often relies on extensive interaction between knowledge engineers and experts. If the purpose of such interaction is to model a multi-agent system, generic models of agents can be used to structure the process of knowledge acquisition. If the purpose of such interaction is to model a specific task, generic models of tasks can be used. During such a process, a *shared* (agreed) *model* of agents or tasks can be acquired, a model on which both the knowledge engineers and the experts agree (Brazier, Jonker, Treur and Wijngaards, 1999*b*). A shared task model is, in fact, a mediating model (Ford, Bradshaw, Adams-Webber, and Agnew, 1993). It mediates between a knowledge engineer and an expert.

Existing models, usually generic models, are most often used to initially structure knowledge acquisition. Which models are used, depends on the initial description of a task or domain: in interaction with one or more experts existing models are examined, discussed, rejected, modified, combined, refined and/or instantiated.

Compositional generic task models provide a means to specify *problem solving methods* (independent of domain ontologies and domain knowledge). Such compositional task models are *generic* in two senses: they are a description of the problem solving method used in the task both at an abstract level and application domain independent. Initial abstract descriptions of tasks can be used to generate a variety of more specific task descriptions through refinement and composition (for which existing descriptions can be employed) in interaction with experts. During knowledge acquisition, knowledge of the application domain itself is also acquired: such application specific knowledge is modelled independently in knowledge structures, and is included in task models by reference to such structures. Knowledge structures are also shared models: models of the domain. Which techniques are used for knowledge elicitation is not predefined. Techniques vary in their applicability, depending, for example, on the situation, the resources, the task, the processes, the type of knowledge on which the knowledge engineer wishes to focus.

Two kinds of refinement of generic models are distinguished. *Specialisation* is a refinement of a generic model in which the compositional process structure is refined. *Instantiation* is a refinement of a generic model in which the compositional knowledge structure is refined.

In this section two generic models are briefly described: a generic model for diagnostic reasoning (on which the first scenario is based) and a generic model for an agent (on which the second scenario is based).

### 4.5.1 Generic model for diagnostic reasoning

Diagnostic reasoning (as described in this thesis) is based on the generic model of diagnostic reasoning (Jonker and Treur, 1999). Domain independent task-related terms can be distinguished within the generic task model for diagnosis: hypotheses and observations (also referred in literature to as symptoms). Both strategic reasoning and object-level reasoning are of importance in this generic model of diagnosis.

Strategic reasoning is involved in the choice of which hypothesis is to be considered first and which observations should be performed.

The generic model of diagnosis described in (Jonker and Treur, 1999) integrates both diagnoses based on causal and anti-causal knowledge. Earlier models for diagnostic reasoning based on anti-causal knowledge can be found in (Treur, 1993), and its use in knowledge acquisition is described in (Brazier, Jonker, Treur and Wijngaards, 1999*b*). A model for diagnostic reasoning based on causal knowledge can also be found in (Brazier, Jonker, Treur and Wijngaards, 1999*b*). *Causal knowledge* is used for derivations which follow the direction of causality: the predicted observable consequences are derived from hypotheses (possible causes), after which (some of) the predicted observations are verified. *Anti-causal knowledge* is used to derive hypotheses from information on observables. This derivation is against the direction of causality: it proceeds from observable findings (in particular, those that actually

were observed) to the causes. The diagnostic strategy employed is left open in the most generic model of diagnostic decision making: this is added in the specialisation of the model.

The generic model for diagnostic reasoning has been employed as an example throughout this chapter and is described in full detail in (Jonker and Treur, 1999). The generic model of diagnostic reasoning has been applied to several domains, e.g., the domain of infant cardiological diagnosis and soil sanitation (Brazier, Jonker, Treur and Wijngaards, 1999*b*; Boelens, 1991). A specialised and instantiated model of diagnosis for the example domain employed in this thesis is later subject of re-design, see Chapter 11.

### 4.5.2 Generic agent model

Agents are often designed to perform their own specific tasks, for example the design of an artefact. In addition, a number of generic agent tasks can be identified. This section describes a generic agent model in which such generic agent tasks are modelled (Brazier, Jonker and Treur, 1996). This model abstracts from the specific task of the agent and domain of application and can be (re)used for a large variety of agents. Instead of designing each and every new agent individually from scratch, a generic agent model can be used to structure the design process: the acquisition of a specific agent model is guided by the generic structures in the model.

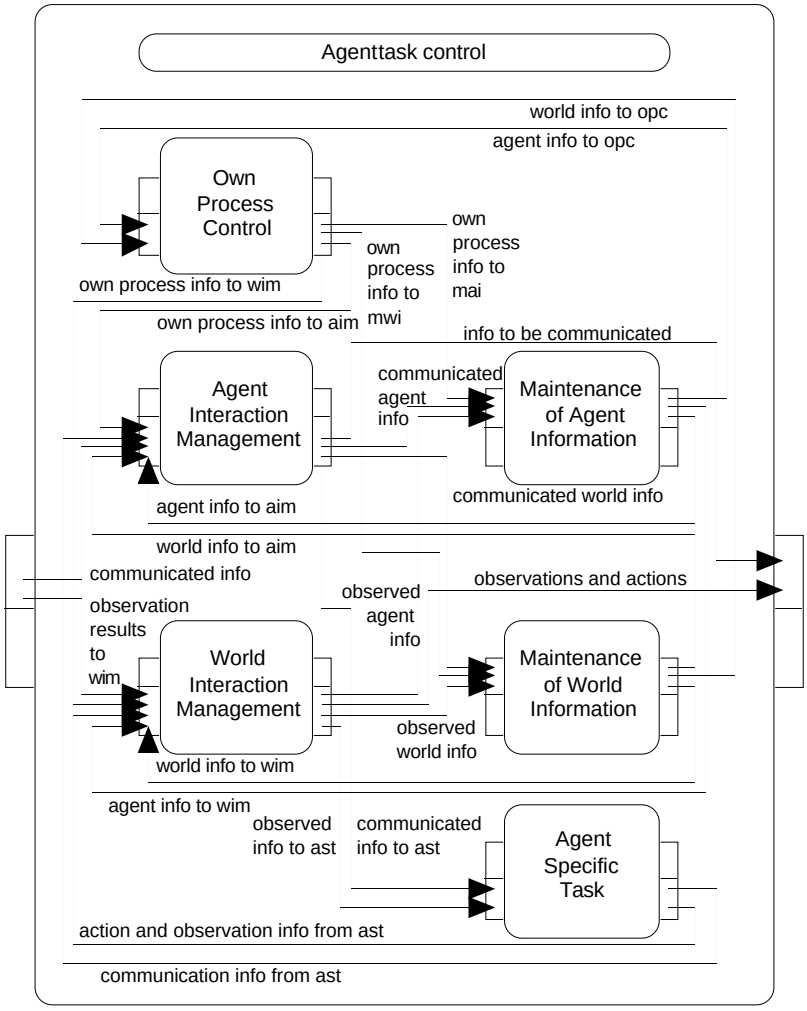


**Figure 4.17** A generic agent model for weak agency.

The composition within an agent capable of reasoning, acting and communicating is shown in Figure 4.17. This agent model supports the notion of weak agent, for which *autonomy*, *pro-activeness*, *reactiveness* and *social abilities* are distinguished as characteristics (Wooldridge and Jennings, 1995). The type of agent model depicted in Figure 4.17:

- reasons about its own processes (supporting *autonomy* and *pro-activeness*),

- communicates with and maintains information about other agents (supporting *social abilities*, and *reactiveness* and *pro-activeness* with respect to other agents), and
- interacts with and maintains information about the external world (supporting *reactiveness* and *pro-activeness* with respect to the external world).

The exchange of information within the generic agent model can be described as follows. Observation results are transferred through the information link observation results to wim from the agent's input interface to the component world interaction management. In addition, the component world interaction management receives belief information from the component maintenance of world information through the information link world info to wim, and the agent's characteristics from the component own process control through the link own process info to wim. The selected actions and observations (if any) are transferred to the output interface of the agent through the information link observations and actions.

The component maintenance of world information receives meta-information on observed world information from the component world interaction management, through the information link observed world info and meta-information on communicated world information (through the link communicated world info) from the component agent interaction management. Epistemic information from maintenance of world information, epistemic world info, is transferred to input belief info on world of the components world interaction management, agent interaction management and own process control, through the information links world info to wim, world info to aim and world info to opc.

Comparably the component maintenance of agent information receives meta-information on communicated information from the component agent interaction management, through the information link communicated agent info and meta-information on observed agent information (through the link observed agent info) from the component world interaction management. Epistemic information, epistemic agent info, is output of the component maintenance of agent information, becomes input belief info on agents of the components world interaction management, agent interaction management and own process control, through the information links agent info to wim, agent info to aim and agent info to opc.

In Chapter 10 the generic agent model described above is specialised for a design agent. In Chapter 12 this design agent is instantiated to re-design a multi-agent system.

## 4.6 Informal and formal semantics

Often textual and graphical notations are used during design of a compositional system. This has a practical use for the (human) designers, as, e.g., graphical representations can be used to convey information. However, these notations need a well-defined semantics shared by its users, otherwise misunderstandings can occur among designers. A formalisation of the semantics is a means to unambiguously record the semantics. An advantage of a formalisation for compositional systems is that supporting tools (e.g., for modelling and verification and validation) can be developed. Generic models of tasks and agents can also be (partially) verified and validated in advance.

Requirements on properties of compositional systems are not only expressed in terms of desired final output of the systems, but also in terms of the behaviour of the systems, e.g., requirements concerning interactions among agents. Complex processes, such as design tasks or multi-agent systems, are often extremely dynamic. The behaviour exhibited by systems modelling such processes is often the result of interaction between processes. A temporal approach is taken to formalise the semantics of a system, so that the changes of information states over time can be modelled. The description of the compositional structure specifies which changes of information states are possible and anticipated, and which behaviour is intended.

The semantical formalisation of a compositional system adopted in this thesis adheres to its compositional structure. A state-based semantics is chosen where each component has an information state. Partial models (Blamey, 1986; Langholm, 1988) are used to formalise information states, representing (incomplete) world descriptions (e.g., Langen and Treur, 1989). To define formal semantics of behaviour in (hierarchical) compositional architectures, a

previously developed approach based on (partial) temporal models is adopted (Engelfriet and Treur, 1994; Gavrila and Treur, 1994; Treur, 1994). Within this approach a trace is formalised by a partial temporal model, i.e., a sequence of partial models. The semantics of a complex process is formalised by a set of (alternative) partial temporal models. The temporal semantics of compositional reasoning systems is defined in detail for the sequential case in (Brazier, Treur, Wijngaards and Willems, 1999) and briefly described in (Brazier, Jonker and Treur, 1998) as included below.

Information types form the language with which ground atoms can be defined. Each component refers to information types (e.g., in its input and output interface). An *information state* M of a component D is an assignment of truth values {true, false, unknown} to the set of ground atoms that play a role within D. The compositional structure of D is reflected in the structure of the information state: an information state is a combination of information states for each of the process abstraction levels. The set of all possible information states of D is denoted by IS(D).

A *trace* $\mathcal{M}$ of a component D is a sequence of information states $(M^t)_{t \in N}$ in IS(D). The set of all traces is denoted by Traces(D). Given a trace $\mathcal{M}$ of component D, the information state of the input interface at time point t of component C within the component D is denoted by $state_D(\mathcal{M}, t, input(C))$, where C is either D or a sub-component of D. Analogously, $state_D(\mathcal{M}, t, output(C))$, denotes the information state of the output interface of component C at time point t of the component D, and $state_D(\mathcal{M}, t, interface(C))$ for both input and output. Given a trace $\mathcal{M}$ of component D, the task control information state of component C at time point t of the component D is denoted by $state_D(\mathcal{M}, t, tc(C))$, where C is either D or a sub-component of D. A specific *behaviour pattern* at the process abstraction level defined by C is described by a trace of the form $state_D(\mathcal{M}, t, interface(C))$ over time. *Behaviour* at the same process abstraction level is described by a set of such traces. Note that behaviour defined in this manner is in principle nondeterministic. The set of traces defining a behaviour are alternatives for the patterns the system can generate.

The temporal semantics of the task control is used as an illustration of the formalisation. The behaviour of a system (i.e., the processes in the kernel of a component) results in a trace of information states. The predicates (defined in a generic manner) in the task control of the encompassing component refer to two of such information states: the *current* information state and the *previous* information state. All conclusions drawn by the task control refer to the *next* information state, i.e., what is to happen. Figure 4.18 depicts this for an example task control knowledge element. As time passes, different information states are produced. The *current information state* is always the latest information state that was produced. This then automatically defines what the previous information state is.

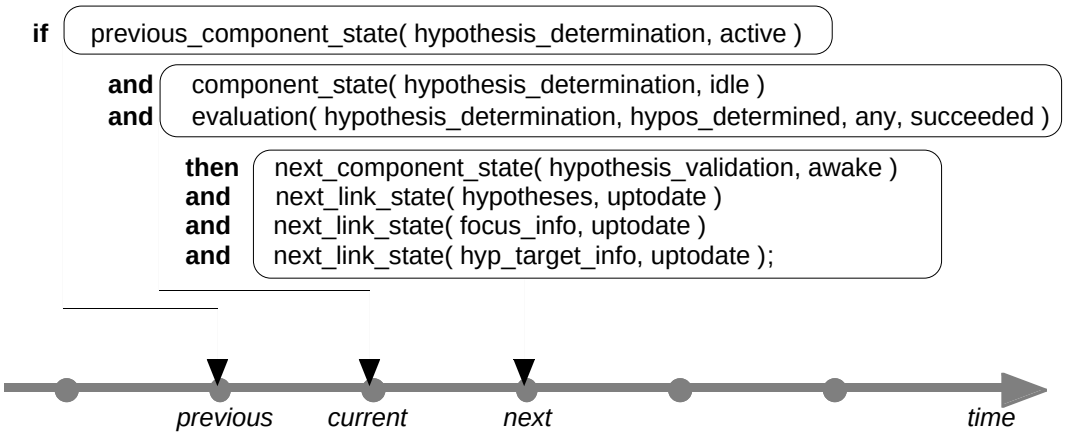


**Figure 4.18** Temporal semantics of a task control knowledge element of the example diagnostic system (see the end of Section 4.2.2).

The statements derived about the next information state restrict which component and information link will become active with particular control settings. Note that no "predictions" are made about the success or failure of a task made active in the next information state.

The above description of semantics defines the meaning of task control for the operationalisation of systems. Notions that can also be found in executable temporal logic Barringer, Fisher, Gabbay, Owens and Reynolds, 1996) are employed in the operationalisation of a compositional structure, thus enabling automatic generation of prototype implementations.

## 4.7 Discussion

In this chapter a structure has been described which can be used to specify compositional systems. This structure has been show to explicitly

- represent processes and process composition, realising 'compositionality of processes',
- represent knowledge structures and knowledge composition, realising 'compositionality of knowledge',
- represent the relationships between processes and knowledge structures, and
- provide a formal semantics for the specification language.

The structure of a compositional system described in this chapter is a substantial extension of the structure as described in (Langevelde, Philipsen and Treur, 1992). Four fundamentally new properties have been added compared to the DESIRE version of 1992. This new version of DESIRE has been developed partly in the context of the research presented in this thesis. These four fundamental new properties are:

- *Compositionality*. Compared to earlier versions, the notion of compositionality of both processes (components) and knowledge has been incorporated. The explicit nesting of components makes it possible to model each task in a task hierarchy (instead of only the primitive tasks). Also task control knowledge is included in this nesting relationship: composed components include task control knowledge: they encapsulate both information and behaviour.
- *Graphical representation*. Both processes and knowledge can be graphically represented, in addition to the textual, concise, notation.
- *Concurrency*. In the new DESIRE, the notion of concurrency is explicitly modelled and specified. That is, components can be activated either sequentially or in parallel (as specified by task control knowledge), whereas the previous version of DESIRE was purely sequential.
- *Agents*. With limited task control knowledge within a component, internal components can be given 'life' after which these components operate autonomously as agents. There is no restriction on the specification of the task control knowledge within these agents.

| *no.* | *Desideratum* | *Explanation* |
|---|---|---|
| ds1 | Unambiguous, formal, representation language. | The textual, concise, representation and graphical representation have an underlying formal definition. |
| ds2 | Formal semantics. | A formal (temporal) semantics is defined for both static and dynamic aspects. |
| ds3 | Explicit representation of both static and dynamic properties. | See Chapter 5. |
| ds4 | Operationalisation. | Within the DESIRE software environment tools are available to automatically generate prototype systems. |
| ds5 | Compositionality as a structuring principle. | A compositional approach has been taken both for processes and knowledge, including an explicit relation between process composition and knowledge composition. |

**Table 4.3** Desiderata on a description of a compositional system and their realisation in the DESIRE modelling framework

The desiderata on a description of a compositional system (identified in Section 2.3) are explicitly fulfilled by (the current) DESIRE, as shown in Table 4.3. The realisation of these desiderata influences the realisation of desideratum dr1 ("representation of a knowledge-intensive system as a design object description").

For a comparison of the current DESIRE with other modelling frameworks for knowledge-intensive systems, see

- (Brazier and Wijngaards, 1997, and 1998 for an extended version) for a comparison on the basis of purposes underlying some modelling frameworks,
- (Schreiber and Birmingham, 1996) for a comparison of applications of modelling frameworks in the domain of elevator configuration.

Briefly compared to approaches in knowledge engineering, the formal specification language of DESIRE has been shown in comparison to other specification languages to be more flexible in modelling reasoning patterns (Harmelen and Fensel, 1995). In terms of expressive power, declarativeness, adequacy to specify dynamic aspects of reasoning patterns, possibility to specify multi-level architectures, adequacy to specify non-classical reasoning, executability and availability of formal semantics, the formal specification framework DESIRE is to some extent comparable (Brazier, Wijngaards, 1997) to other formal specification frameworks such as CommonKADS/(ML)$^2$ (Harmelen and Balder, 1992) and MIKE/KARL (Fensel, 1995). It differs in that specifications are executable and agents and integrated systems can be specified. The formal specification languages differ in expressiveness of control knowledge (see also Treur and Wetter, 1993; Harmelen and Fensel, 1995; REVISE, 1996).

Within the agent community formal agent modelling languages are rare. Two of such languages, viz. Concurrent METATEM (Fisher, 1993; 1994) and DESIRE, have been compared in (Mulder, Treur and Fisher, 1997). The dynamic behaviour of simpler agents is less concisely expressed in DESIRE than agents described in a temporal specification in Concurrent METATEM. In contrast to Concurrent METATEM, the compositional approach in DESIRE provides structure for modelling larger and more complex agents.

The structure of a compositional system described in this chapter is used in the remainder of this thesis. In the next chapter properties of compositional systems are discussed.

# 5 Compositional Systems: Properties

Requirements play an important role in a process of design: they guide the direction in which solutions are sought and determine on which properties the results of the design process will be evaluated. Requirements on compositional systems may be formulated in terms of required properties. Such properties may refer to static aspects of a compositional system (i.e., the structure), or dynamic aspects of a compositional system (i.e., the behaviour), or both.

In Chapter 2, a number of desiderata have been formulated. One of these desiderata, ds3 ("explicit representation of both static and dynamic properties"), is directly related to properties of compositional systems. An ontology is a means to explicitly represent such properties. This chapter describes how properties, relations between properties, and relations between properties and structures may be represented. A number of properties and relations for two kinds of compositional systems: a diagnostic reasoning system and a multi-agent system, are used to illustrate the approach taken.

Section 5.1 describes properties of compositional systems in general. Some properties and relations between these properties of compositional systems in the two application domains are illustrated in Section 5.2 (properties of compositional diagnostic reasoning systems) and Section 5.3 (properties of compositional multi-agent systems). In Section 5.4 a relation between properties of compositional systems and verification of these systems is addressed. Section 5.5 concludes this chapter with a brief discussion.

## 5.1 Properties of compositional systems

In this thesis, the terms *abilities* and *properties* are both employed (Brazier, Jonker, Treur and Wijngaards, 1998*b*). A distinction is made between agents and the external world: the external world is not considered to be an agent. As a result for the external world only the term property is used. The same holds for the multi-agent system as a whole.

In the knowledge engineering community in general, one of the areas of research is the characterisation of tasks and problem solving methods: the identification of *capabilities* of problem solving methods, e.g., see (Benjamins, 1993; Fensel, 1995; 1997; Fensel, Motta, Decker and Zdrahal, 1997; Harmelen and Teije, 1998; Orsvärn, 1996; Wielinga, Schreiber and Breuker, 1992; Breuker, 1997). Another related characterisation of problem solving methods is based on so-called *assumptions*, e.g., see (Benjamins, Fensel and Straatman, 1996; Fensel and Straatman, 1998). These assumptions make the dependencies between various parts of a problem solving method explicit.

Another area of research pursued by the knowledge engineering community is the identification of the correspondence between system structures and properties (or generic descriptions) (Beys, Benjamins, and van Heijst, 1996). Generic descriptions are employed for various reasons, including building, maintaining and indexing libraries of e.g., reusable problem solving methods (Benjamins, 1993; Aben, 1995; Benjamins and Aben, 1997).

Generic properties may state characteristics such as 'output correctness', while task-related more specific properties may state characteristics such as 'capable of performing diagnostic reasoning'. Task-specific vocabularies provide a means to formulate properties in terms of a task, thereby providing a richer vocabulary (but less applicable in general (e.g., Motta and Zdrahal, 1998).

The term *property* is more general than the term *ability*; when characterising (parts of) processes in this thesis, the term *property* is employed. The term property subsumes abilities, capabilities, and assumptions of (parts of) knowledge-intensive systems.

A graphical notation for refinement relationships between properties is shown in Figure 5.1. A distinction between 'combining' and 'or' refinements is shown. A 'combining'

refinement indicates, when read from left to right, that property ‘x’ is refined into properties ‘y1’, ‘y2’*and* the property ‘is capable of combining y1 and y2’. An ‘or’ refinement indicates, when read from left to right, that property ‘x’ is refined by property ‘y1’, or by property ‘y2’, or by the combination of properties ‘y1’ and ‘y2’.

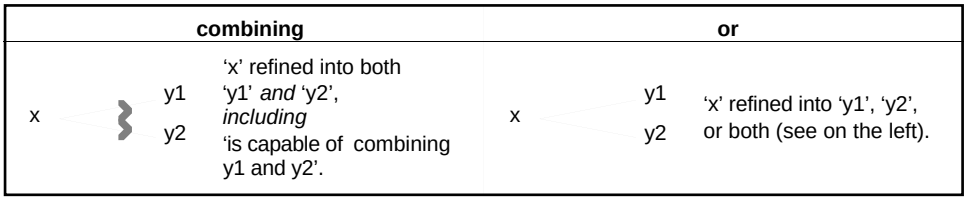


**Figure 5.1** Refinement graph legend.

When read from right to left, a ‘combining’ refinement indicates that when properties ‘y1’, ‘y2’, and ‘is capable of combining y1 and y2’ are present, then property ‘x’ is also present. An ‘or’ refinement indicates, when read from right to left, that if property ‘y1’ is present, property ‘x’ is also present, and that if property ‘y2’ is present, property ‘x’ is also present.

In the next two sections properties for the example diagnostic reasoning system and the example multi-agent system are discussed in some detail.

## 5.2 Properties for the example diagnostic system

This section describes an ontology for properties of the example diagnostic system, described in Section 3.3 Refinement-relations are defined for the properties discussed in this section. The top-level property for the diagnostic reasoning system is the property is capable of diagnostic reasoning.

An ontology for properties of a diagnostic system provides a means to express properties with respect to overall behaviour of a system. Two examples of such properties are the system proposes fewer hypotheses (in comparison to random proposal), and the system is capable of explicit strategic reasoning for determination of hypotheses.

The following properties are distinguished for the example diagnostic reasoning system:

- properties related to the property is capable of diagnostic reasoning (see Section 5.2.1), and
- refined properties related to the property is capable of strategy determination (see Section 5.2.2).

An example of the deduction of a property on the basis of a specific diagnostic system is given in Section 5.4.

### 5.2.1 Properties of diagnostic reasoning

A number of refined properties related to the property is capable of diagnostic reasoning, can be distinguished. The property *is capable of diagnostic reasoning* is a property of the entire process, to which other properties can be related; the property is capable of diagnostic reasoning can be refined as shown in Figure 5.2.

Three more specific properties related to the property of is capable of diagnostic reasoning, are

- is capable of determination of hypotheses,
- is capable of validation of hypotheses, and
- is capable of combining determination of hypotheses and validation of hypotheses.

Five more specific properties related to the property is capable of determination of hypotheses, are

- is capable of generation of hypotheses,
- is capable of comparison of hypotheses,
- is capable of selection of hypotheses, and
- is capable of combining generation of hypotheses, comparison of hypotheses, and selection of hypotheses.

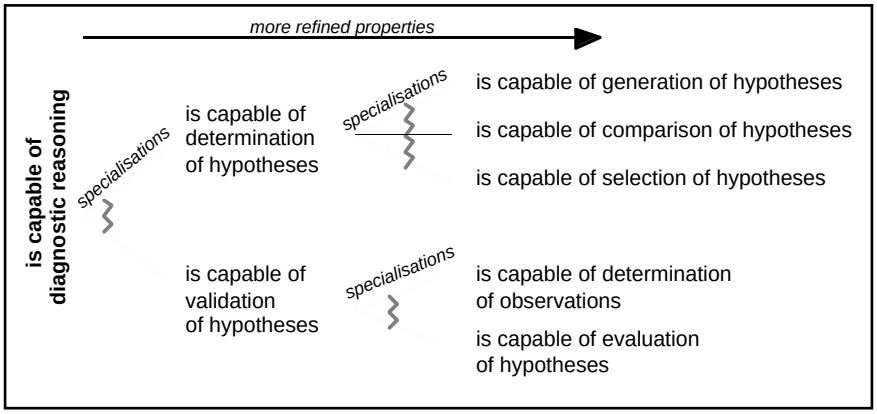


**Figure 5.2** Refinement of the property is capable of diagnostic reasoning.

The property is capable of validation of hypotheses can be refined into

- is capable of determination of observations,
- is capable of evaluation of hypotheses, and
- is capable of combining determination of observations and evaluation of hypotheses.

The generic model of diagnostic reasoning, introduced in Section 3.3 and shown in more detail in Chapter 4, was designed with these a number of these properties as requirements. The process composition in the generic model is based on the refinements of the property is capable of diagnostic reasoning. These properties are a means to formulate requirements on a diagnostic reasoning system, as shown in Chapter 11.

### 5.2.2 Properties related to strategy determination for hypothesis determination

The determination of strategies for hypothesis determination needs to be integrated with other processes involved in the determination of hypotheses. Therefore, the property is capable of integrating a strategy in hypothesis determination is a necessary refinement. Whether strategies are determined autonomously or in interaction with a user, is a choice. The refinements in Figure 5.3 reflect this notion of refining strategy determination with other properties.

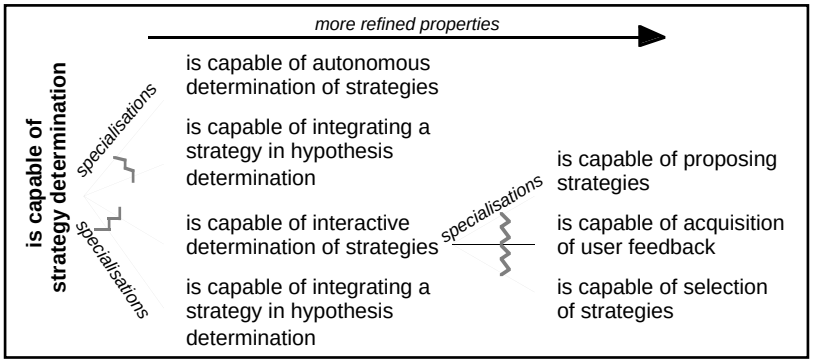


**Figure 5.3** Refinement of the property is capable of strategy determination.

Two sets of specific properties related to the property is capable of strategy determination, are

- is capable of integrating a strategy in hypothesis determination,
- is capable of interactive determination of strategies, and
- is capable of combining interactive determination of strategies and integrating a strategy in hypothesis determination;

and

- is capable of integrating a strategy in hypothesis determination of strategies,
- is capable of autonomous determination of strategies, and

- is capable of combining autonomous determination of strategies and integrating a strategy in hypothesis determination.

Four more specific properties related to the property of interactive determination of strategies, are

- is capable of proposing strategies,
- is capable of acquisition of user feedback,
- is capable of selection of strategies, and
- is capable of combining proposing strategies, acquisition of user feedback, and selection of strategies.

## 5.3 Properties for the example multi-agent system

A multi-agent system (MAS) is composed of interacting agents and the external world. Properties can be assigned to

- a multi-agent system as a whole,
- individual agents,
- the external world,
- an individual agent in relation to the agents and the world with which it interacts,
- the world in relation to the agents with which it interacts.

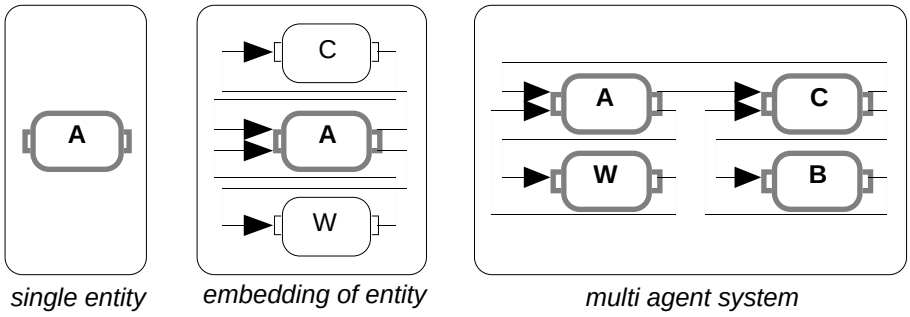


**Figure 5.4** Aggregation levels within a multi-agent system consisting of agents A, B, and C, and external world W.

Three different levels of aggregation can be distinguished, as shown in Figure 5.4: *single entity*, *embedded entity* and *multi-agent system*. The properties assigned to each level are characterised as follows:

| | |
|---|---|
| *single entity* | An 'agent' or 'external world' pur sang: what it can do, can't do, abilities or properties. Formulated in terms of an agent or external world. |
| *embedded entity* | Abilities and properties of a single agent in relation to other agents and/or the external world with which it interacts. Abilities and properties of the external world in relation to the agents with which it interacts. This includes abilities relating to communication between agents and/or interaction of agents with the external world. Examples of the embedding relationship for Figure 5.4 include: embedding_of( A ) = { A, C, W }, embedding_of( B ) = { B, C }, and embedding_of( W ) = { W, A }. |
| *multi-agent system* | Properties in terms of the entire MAS, formulated in terms of the MAS, or 'sets of agents', or 'sets of agents and the external world'. |

Note that the aggregation levels do not have an 'inclusion' relationship, i.e., abilities or properties are not automatically 'inherited'. However, abilities and properties at one level, may be related to abilities and properties at another level.

In this section an ontology of properties for multi-agent systems is described. The properties of a multi-agent system are outlined in Section 5.3.1, after which the abilities and properties of single entities are described: the generic abilities of a single agent are discussed in Section 5.3.2, and the properties of the external world are addressed in Section 5.3.3. An example of how a property for a specific agent can be shown to hold is given in Section 5.4

### 5.3.1 Properties of a multi agent system

The properties of a multi-agent system, although related to the abilities of individual agents and the properties of an external world, are properties described at the level of the multi-agent system itself. A property of a multi-agent system is, for example, the property is capable of distributed information gathering by two agents (possibly in interaction with the external world). Properties of a multi-agent system relate to the roles agents fulfil in relation to each other.

As a more detailed example, consider the multi-agent system in Figure 5.4. This multi-agent system *could* be capable of following a specific distributed problem-solving method in which an agent is required which gathers information upon request for another agent. For example agent D gathers information in the world for agent A: agent D provides answers to queries from agent A by making observations in the external world W. Agent D can be considered to be an information gatherer for agent A. The initial multi-agent system, however, does not include agent D and therefore does not satisfy the property is capable of distributed information gathering. In Chapter 12 this example is extended by illustrating how the multi-agent system can be re-designed to obtain another multi-agent system which does have this property.

The property is capable of distributed information gathering can be refined into four more specific properties as shown in Figure 5.5: an agent has the property is capable of request initiation, an agent has the property is capable of information gathering, an external world has the property is capable of information provision, and the multi-agent system has the property is capable of combining request initiation, information gathering and information provision. In Figure 5.5 properties residing at all levels of aggregation are depicted, from left to right: multi-agent system, embedded entity, and single entity.

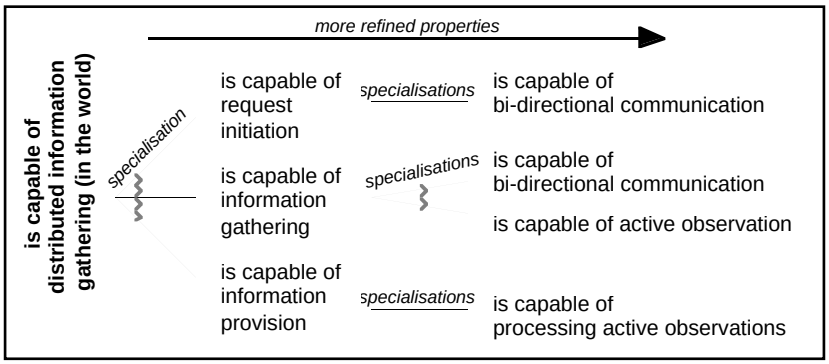


**Figure 5.5** Refinement of property is capable of distributed information gathering.

For entities in the multi-agent system with the refinements of request initiation, information gathering, and information provision additional, more specific properties can be identified (as shown in Figure 5.5). The specific ability related to the property is capable of request initiation, is the ability is capable of bi-directional communication. The three specific abilities related to the property is capable of information gathering role are is capable of bi-directional communication, is capable of active observation, and is capable of combining bi-directional communication and active observation. Related to the property is capable of information provision, a property of the external world, is the property is capable of processing active observations.

The properties of agents and the external world are described in more detail in the next two sub-sections.

### 5.3.2 Properties of an agent

Individual agents often have different abilities and properties. Properties of agents, which are not naturally called abilities, are, for example, properties concerning safety (e.g., an agent will never …). As an example of a safety property, consider the first of the Three Laws of Robotics, proposed by Asimov (1963, reprint: 1991): "A robot may not injure a human being or, through inaction, allow a human being to come to harm". Such properties are not addressed in this thesis. A number of abilities of an agent are the following

- is capable of bi-directional communication,
- is capable of co-operation,
- is capable of agent own process control, and
- is capable of world interaction.

Abilities can be refined, both with respect to their *specialisation* (refinement of the ability into more specific abilities) and with respect to their *realisation* (refinement of the ability into more fine-grained abilities related to reasoning about the domain referred to by the first ability, and more fine-grained abilities related to the effectuation of the ability).

The specialisation relationship and realisation relationship each define an 'implication' relationship. If the more specific abilities exist within an agent, then the ability for which these specific abilities are a specialisation also exists. This relationship is depicted by 'combining' and 'or' notations in the refinement relationships for abilities and properties (see Figure 5.1 for a description of the legend of the notations used).

**Bi-directional communication.** Figure 5.6 shows the refinement relationships for the ability is capable of bi-directional communication. The more specific abilities related to is capable of bi-directional communication are the ability to communicate to others (is capable of unidirectional communication *to* others), to receive communication from others (is capable of unidirectional communication *from* others), and the ability to combine unidirectional communication to and from others (is capable of combining unidirectional communication to and from others). The abilities related to the realisation of the ability of bi-directional communication are the ability is capable of *reasoning* about bi-directional communication, the ability is capable of *executing* bi-directional communication, and the ability is capable of combining reasoning about bi-directional communication and executing bi-directional communication.

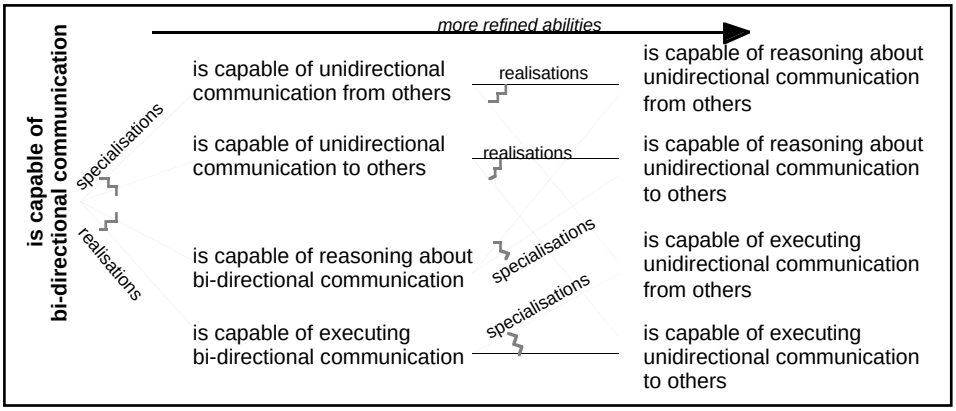


**Figure 5.6** Refinement of the ability is capable of bi-directional communication.

These more specific abilities are further refined, and related to the ability is capable of reasoning about unidirectional communication from others, the ability is capable of reasoning about unidirectional communication to others, the ability is capable of executing unidirectional communication from others, and the ability is capable of executing unidirectional communication to others, and four properties which combine the previous four abilities (as shown in Figure 5.6):

- is capable of combining reasoning about unidirectional communication from others and executing unidirectional communication from others,

- is capable of combining reasoning about unidirectional communication to others and executing unidirectional communication to others,
- is capable of combining reasoning about unidirectional communication from others and reasoning about unidirectional communication to others, and
- is capable of combining executing unidirectional communication from others and executing unidirectional communication to others.

**Co-operation.** Figure 5.7 shows the refinement relationships for the ability is capable of co-operation. The more specific abilities related to is capable of co-operation are, for this example multi-agent system, the ability is capable of *planning* co-operation, the ability is capable of *monitoring* co-operation, the ability is capable of own process control, the ability is capable of bi-directional communication, and the ability is capable of combining planning co-operation, monitoring co-operation, own process control, and bi-directional communication. The abilities related to the realisation of the ability is capable of co-operation are the ability is capable of *reasoning* about co-operation, the ability is capable of *executing* co-operation, the ability is capable of own process control, the ability is capable of bi-directional communication, and the ability is capable of combining reasoning about co-operation, executing co-operation, own process control, and bi-directional communication. The refinements of the co-operation ability defines a strong notion of co-operation by specifying the integration of co-operation with agent own process control and bi-directional communication. Weaker notions of co-operation are also possible. These notions are not investigated in this thesis.

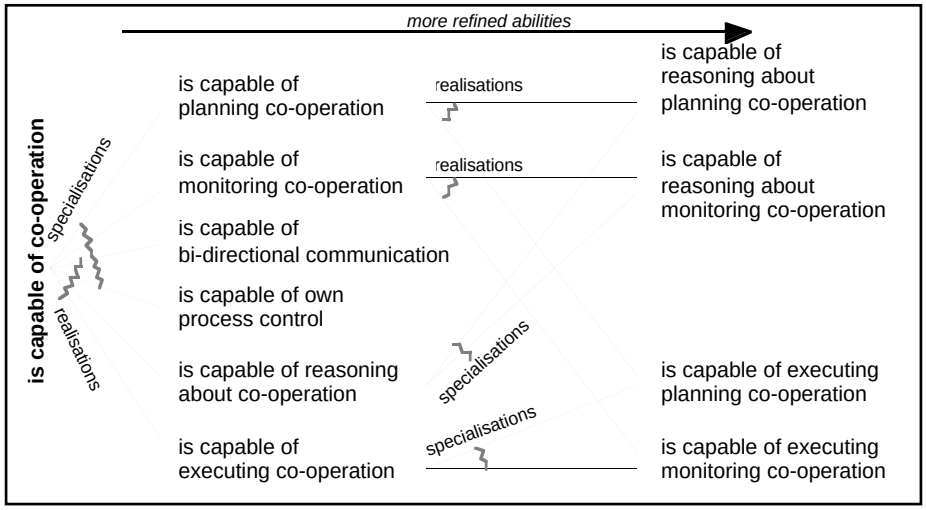


**Figure 5.7** Refinements of the ability is capable of co-operation.

The abilities related to the realisation of the ability is capable of planning co-operation are the ability is capable of reasoning about planning co-operation, the ability is capable of executing planning co-operation, and the ability is capable of combining reasoning about planning co-operation and executing planning co-operation. The abilities related to the realisation of the ability is capable of monitoring co-operation are, likewise, the ability is capable of reasoning about monitoring co-operation, the ability is capable of executing monitoring co-operation, and the ability is capable of combining reasoning about monitoring co-operation and executing monitoring co-operation. These realisation related abilities are, in fact, specialisations of the abilities related to the realisation of the ability is capable of co-operation. The ability is capable of reasoning about planning co-operation, the ability is capable of reasoning about monitoring co-operation, and the ability is capable of combining reasoning about planning co-operation and reasoning about monitoring co-operation are refinements of the ability is capable of reasoning about co-operation. The ability is capable of executing planning co-operation, the ability is capable of executing monitoring co-operation, and the ability is capable of combining executing planning co-operation and executing monitoring co-operation are refinements of the ability is capable of executing co-operation.

**Agent own process control.** The ability of an agent to be able to control its own processes can be refined as shown in Figure 5.8.

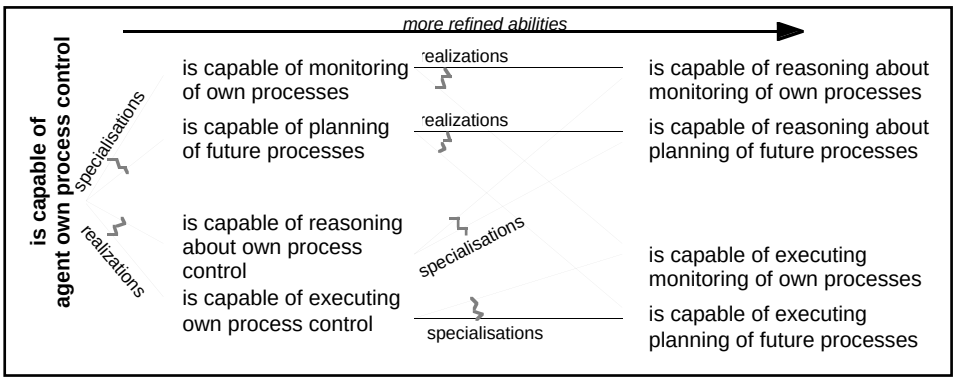


**Figure 5.8** Refinement of the ability is capable of agent own process control.

Three more specific abilities related to the ability is capable of own process control, are

- is capable of *monitoring* of own processes,
- is capable of *planning* of future processes, and
- is capable of combining monitoring of own processes and planning of future processes.

These abilities are each further refined to abilities in which realisation (reasoning and execution) is explicitly defined for each of these abilities.

The abilities related to the realisation of the ability of an agent to control its own processes are the ability of an agent to *reason* about its own process control, the ability of an agent to *execute* its own process control, and the ability of an agent to *combine* reasoning about own process control and executing own process control. For a number of these abilities two more specific abilities are depicted. Reasoning about an agent's own process control is related to the abilities to reason about current and future process control, and to combine reasoning about current and future process control. Executing an agent's own process control is related to an agent's abilities to execute current processes monitoring, to execute future processes planning, and to combine executing current process monitoring and executing future process planning.

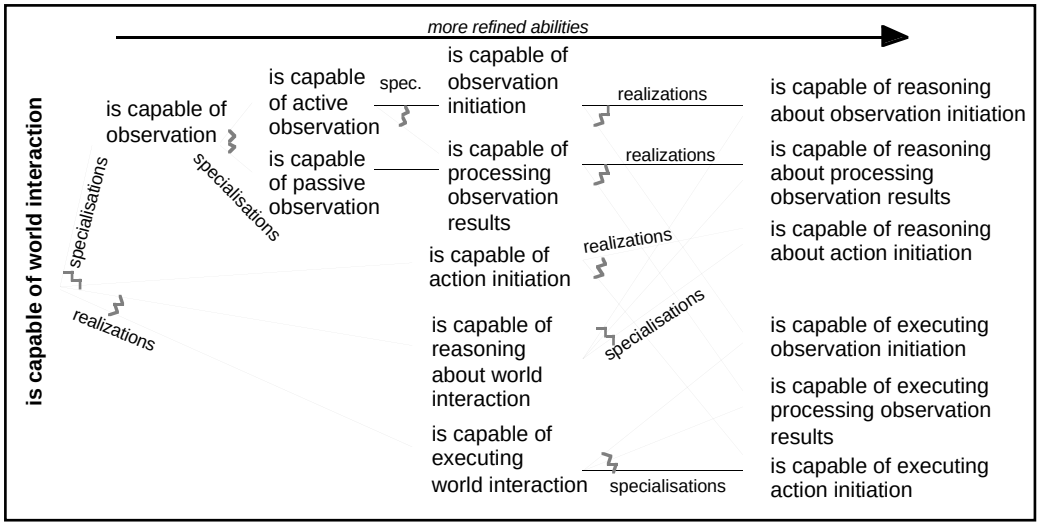


**Figure 5.9** Refinement of the ability is capable of world interaction.

**World interaction.** The ability to interact with the external world can be refined to more specific abilities: the ability is capable of observation in the world, the ability is capable of initiation of actions in the world, and the ability is capable of combining observation and initiation of actions,

as shown in Figure 5.9. The ability is capable of observation is further refined into the ability is capable of passive observation, the ability is capable of active observation, and the ability is capable of combining passive observation and active observation. The ability is capable of active observation is refined into three abilities: the ability is capable of observation initiation, the ability is capable of processing observation results, and the ability is capable of combining observation initiation and processing observation results. The ability of passive is capable observation is refined into the ability is capable of processing observation results. These abilities are each further refined to abilities in which realisation (reasoning and execution) is explicitly defined for each of these abilities. A strong notion of world interaction is defined by these refinements. Weaker notions are possible, but are not employed in this thesis.
The abilities related to the realisation of the ability of an agent to interact with the world are the ability of an agent to *reason* about its interaction with the world, the ability of an agent to *execute* interaction with the world, and the ability of an agent to *combine* reasoning about world interaction and executing world interaction. For each of these abilities four more specific abilities are depicted. Reasoning about world interaction is related to the abilities to reason about observation initiation, to reason about processing observation results, to reason about action initiation, and to combine reasoning about observation initiation and reasoning about processing observation results and reasoning about action initiation. Executing interaction with the external world is related to an agent's abilities to execute observation initiation, to execute processing observation results, to execute action initiation, and to combine executing observation initiation and executing processing observation results and executing action initiation.

### 5.3.3 Properties of an external world

The external world, part of a multi-agent system, has properties (not called abilities). These properties are related to how "equipped" the world is to handle interaction with agents.

*Properties* of external world are depicted in Figure 5.10. The property is capable of world interaction (from the point of view of the external world) can be realised by three properties: the property is capable of processing observations, the property is capable of processing actions, and the property is capable of combining processing observations and processing actions. These properties are the counterpart of the agent *ability* of world interaction: the *ability* is capable of active observation and the *ability* is capable of passive observation.

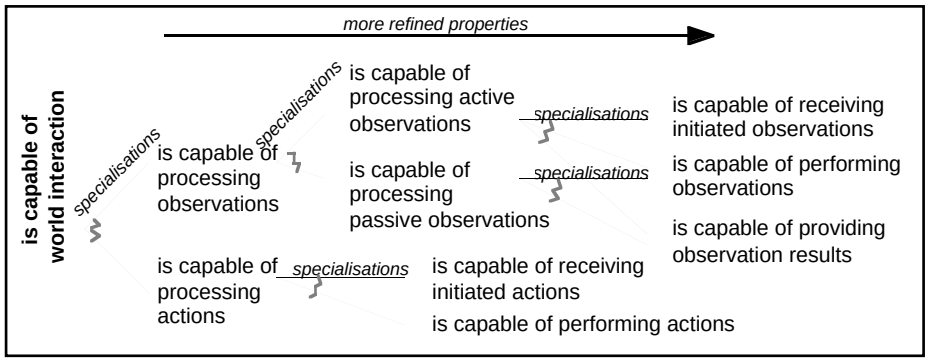


**Figure 5.10** Refinement of the property is capable of world interaction.

The property of processing observations can be refined into the more specific properties:

- is capable of processing active observations (i.e., counterpart for the ability is capable of active observation),
- is capable of processing passive observations (i.e., counterpart for the ability is capable of passive observation), and
- is capable of combining processing active observations and processing passive observations.

The property of processing actions can be refined into the properties:

- is capable of receiving initiated actions,
- is capable of performing actions, and
- is capable of combining receiving initiated actions and performing actions.

The property is capable of processing active observations can be refined into the property is capable of receiving initiated observations, the property is capable of performing observations, the property of is capable of providing observation results, and the property is capable of combining receiving initiated observations, performing observations, and providing observation results. The property is capable of processing passive observations can be refined in to the property is capable of performing observations, the property is capable of providing observation results, and the property is capable of combining receiving initiated actions and performing actions.

As all of these refinements are about the *specialisation* of a property, there is no refinement for reasoning and execution as these refinements are only applicable to agents. This property of the external world can be weakened by, e.g., modifying 'combining'-relationships into 'or'-relationships. Weaker notions of world interaction are not the subject of this thesis.

## 5.4 Verification of properties of a compositional system

The manner in which the properties described in the previous sections can be given formal semantics is not addressed in this thesis. The way in which formalisation of the semantics of properties of compositional systems is possible in terms of temporal semantics, can be found in work on compositional verification of knowledge-intensive systems, for diagnostic systems see (Cornelissen, Jonker and Treur, 1997), for co-operative information gathering (i.e. co-operation of multiple information gathering agents) see (Jonker and Treur, 1998*c*), and for negotiating agents see (Brazier, Cornelissen, Gustavsson, Jonker, Lindeberg, Polak and Treur, 1998). The compositional verification method is based on the process composition of the system being verified, providing a structure for 'proof hiding' and proof composition.

The properties described in (Jonker and Treur, 1998*c*) on co-operative information gatherers concern, among others, the success of co-operation: which combinations of reactiveness and pro-activeness are needed for successful co-operative information gathering. In this thesis the example multi-agent system entails co-operation between an initiator of requests for information and *one* information gatherer, not co-operative information gatherers.

The precise semantic formalisation of the properties introduced in this chapter is beyond the scope of this thesis. In contrast to the literature mentioned above, in this thesis an important issue is how properties of a given compositional system can be derived automatically from its (formal) specification. This is achieved by explicit knowledge.

Two instances of knowledge are given below with which the presence of a property can be determined for a given compositional system. In these examples the properties is capable of executing observation initiation and is capable of determination of hypotheses are deduced from the structure of the compositional system. These examples depict how primitive properties can be deduced for a given compositional system, and how properties can be deduced from previously deduced (possibly primitive) properties.

An instance of knowledge to ascertain the presence of the ability of executing observation initiation is given below.

**Example Deducing the ability of executing observation initiation**

The knowledge elements below have been written in semi-formal notation, as the representation of a compositional system *within* another compositional system has not yet been discussed.

The first knowledge element specifies the deduction of the property is capable of executing observation initiation. In Figure 5.11 the relevant structure of the compositional system is sketched. The second knowledge element specifies the deduction of the property is capable of active observation.

**if** a component X in the multi-agent system is characterised as an agent,
**and** the multi-agent system has task control knowledge which makes component X awake,
**and** component X has an information type Xin in its input interface,

**and** information type Xin is characterised to contain ‘observation results’ information,
**and** component X has an information type Xout in its output interface,
**and** information type Xout is characterised to contain ‘observations to be performed’ information,
**and** component X has a sub-component Y,
**and** component Y is characterised as a world interaction management component,
**and** component Y has an information type Yin in its input interface,
**and** information type Yin is characterised to contain ‘observation results’ information,
**and** component Y has an information type Yout in its output interface,
**and** information type Yout is characterised to contain ‘observations to be performed’ information,
**and** component Y uses a knowledge base K,
**and** knowledge base K is characterised to contain ‘observation initiation’ knowledge,
**and** knowledge base K refers to the information types Yin and Yout,
**and** component X has an information link L1 from its input to the input of component Y,
**and** information link L1 transfers information from information type Xin at the input interface of component X to the information type Yin at the input interface of component Y,
**and** component Y has an information link L2 from its output to the output of component X,
**and** information link L2 transfers information from information type Yout at the output interface of component Y to the information type Xout at the output interface of component X,
**and** component X has task control knowledge which makes component Y awake and the information links L1 and L2 awake as well,
**and** a component W in the multi-agent system is characterised as an external world,
**and** the multi-agent system has task control knowledge which makes component W awake,
**and** component W has an information type Win in its input interface,
**and** information type Win is characterised to contain ‘observations to be performed’ information,
**and** component W has an information type Wout in its output interface,
**and** information type Wout is characterised to contain ‘observation results’ information,
**and** the multi-agent system has an information link L3 from the output of component X to the input of component W,
**and** information link L3 transfers information type Xout at the output interface of component X to information type Win at the input interface of component W,
**and** the multi-agent system has an information link L4 from the output of component W to the input of component X,
**and** information link L4 transfers information type Wout at the output interface of component W to information type Xin at the input interface of component X,
**and** the multi-agent system has task control knowledge which makes the information links L3 and L4 awake,

**then** component X ‘is capable of executing observation initiation’.

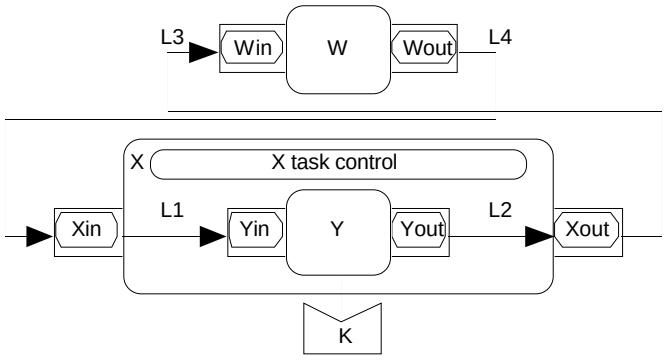


**Figure 5.11** Relevant part of a multi-agent system to deduce the property executing observation initiation.

The knowledge element below illustrates the use of the 'specialisation' refinement relation for the property is capable of active observation as defined in Figure 5.9.

**if** a component X has sub-components Y1, Y2, and Y2,
**and** component Y1 'is capable of observation initiation',
**and** component Y2 'is capable of processing observation results',
**and** component X 'is capable of combining initiation of observations and processing of observation results',
**then** component X 'is capable of active observation'.

An instance of knowledge to ascertain the presence of the property of determining hypotheses is given below.

**Example Deducing the property of is capable of determination of hypotheses**

These knowledge elements have also been written in semi-formal notation, as the representation of a compositional system *within* another compositional system has not yet been discussed.

The first knowledge element specifies the deduction of the property is capable of generation of hypotheses, the second knowledge element specifies the deduction of the property is capable of determination of hypotheses. The second knowledge element illustrates the use of the 'specialisation' refinement relation, read from right to left in Figure 5.2: on the basis of established properties, additional properties can be deduced.

**if** component X is a sub-component of component S,
**and** component X has information types Xin1 and Xin2 in its input interface,
**and** information type Xin1 is characterised to contain 'assessed hypotheses' information,
**and** information type Xin2 is characterised to contain 'interpreted observation results' information,
**and** component X has an information type Xout in its output interface,
**and** information type Xout is characterised to contain 'generated hypotheses' information,
**and** component X is characterised as a 'generation of hypotheses' component,
**and** component X uses a knowledge base K,
**and** knowledge base K refers to the information types Xin1, Xin2, and Xout,
**and** knowledge base K is characterised to contain 'generation of hypotheses' knowledge,
**then** component X 'is capable of generation of hypotheses'.

**if** a component X has sub-components Y1, Y2, and Y3,
**and** component Y1 'is capable of generation of hypotheses',
**and** component Y2 'is capable of comparison of hypotheses',
**and** component Y3 'is capable of selection of hypotheses',
**and** component X 'is capable of combining generation of hypotheses, comparison of hypotheses, and selection of hypotheses',
**then** component X 'is capable of executing observation initiation'.

## 5.5 Discussion

In this chapter properties of compositional systems and knowledge relating to properties are introduced. Ontologies of properties for the example diagnostic reasoning system and the example multi-agent system have been introduced. Knowledge on these properties has been provided: refinements (specialisations and realisations) of abilities and properties have been outlined.

For the example diagnostic reasoning system the following properties have been shown to be related to the property is capable of diagnostic reasoning:

- is capable of determination of hypotheses,
- is capable of generation of hypotheses,
- is capable of selection of hypotheses,
- is capable of comparison of hypotheses,
- is capable of combining generation of hypotheses, comparison of hypotheses, and selection of hypotheses,

- is capable of validation of hypotheses,
- is capable of combining determination of hypotheses and validation of hypotheses,
- is capable of determination of observations,
- is capable of evaluation of hypotheses, and
- is capable of combining determination of observations and evaluation of hypotheses.

The abilities of agents described in this chapter support the notion of weak agency (e.g., Wooldridge and Jennings, 1995).

For the example multi agent system the following abilities and properties have been introduced:

- is capable of co-operation,
- is capable of bi-directional communication,
- is capable of agent own process control,
- is capable of world interaction,
- is capable of world interaction with agents (a property of the external world), and
- is capable of distributed information gathering by several agents (a property of a multi-agent system).

The properties distinguished in this chapter are specific in the sense that they apply to a (example) diagnostic reasoning system and a (example) multi-agent system; more general properties (e.g., correctness of output) have not been included in this chapter. However, the properties and knowledge on relations between properties addressed in this chapter are of importance for the manipulation of requirements (which require these properties) and verification and validation of systems.

Desiderata dr1, dr2, dr3, dr4, ds3 are concerned with properties of compositional systems. According to desideratum ds3 an ontology is needed to represent properties. This desideratum has been realised as is extensively shown in this chapter (the properties and their refinement relations). The desiderata dr1 and dr2 ("representation of a knowledge intensive system as a design object description" and "representation of qualified requirements on compositional systems") require the identification of properties of compositional systems and a means to represent properties and knowledge on properties. The basis for this has been shown in this chapter; the precise formulation of knowledge on properties in the context of the re-design model is shown in Chapters 7 and 9. Desideratum dr4 ("model of the manipulation of requirements on compositional systems") requires the representation of refinement knowledge on properties, examples of which have been given in Section 5.4. The desideratum dr3 ("model of the manipulation of compositional system structures") requires that properties need to be determined on the basis of the structure of a compositional system, examples of which have been given in Section 5.4.

The work in this chapter is related to the properties distinguished with respect to problem solving methods (Benjamins, Fensel and Straatman, 1996; Breuker, 1997; Cornelissen, Jonker and Treur, 1997; Fensel, Motta, Decker and Zdrahal, 1997; Teije and Harmelen, 1994). Within the field of Knowledge Engineering properties of problem solving methods are used to support knowledge engineers during the design process: providing a means to describe existing generic components that may be used, modified or refined during a design process, depending on their applicability in a given situation. The Knowledge Engineering community has not focussed on abilities and properties of agents and their interaction, as addressed in this chapter.

An approach to describe the correspondence between part of a compositional structure and a property in gradual terms is presented in (Harmelen and Teije, 1998): gradual satisfaction of requirements. Using refinements of requirements, and assuming that fulfillment of refinements implies fulfillment of the requirement, provides another form of gradual satisfaction of a requirement (which has refinements). If one or more of the refinements of a requirement are not satisfied, the requirement can be said to be 'partially' satisfied, and it is also precisely known which refinements are not satisfied, an aspect which confounds the interpretation and comparison of gradual satisfaction of requirements.

Within the Knowledge Engineering community work has been done on verification and validation of models, based on properties (not task-specific properties, yet) of problem-solving methods without taking into account the compositional nature of their models. An approach in

which verification and validation of compositional models results in a compositional proof (of diagnostic systems: Cornelissen, Jonker and Treur, 1997; of co-operative agents: Jonker and Treur, 1998*c*; of negotiating agents: Brazier, Cornelissen, Gustavsson, Jonker, Lindeberg, Polak, Treur, 1998), offers an advantage: properties over composed components are assumed to hold, and proven within that composed component by reference to its internal components. This approach prevents the occurrence of very large proofs at the top level of the system in which all processes at all levels of abstraction are incorporated.

Properties of software systems also play a role in re-use and maintenance of software systems. In a survey on software reuse, Krueger (1992) notes that to successfully generate applications, meta-programming is used: knowledge and definitions are employed to, for example, recognise an appropriate domain, define boundaries of the domain and defining underlying abstractions. Such definitions and knowledge can be expressed in terms of properties of software systems.

To illustrate the role these properties can play in re-design, two re-design processes are described in Chapters 11 and 12. A diagnostic reasoning system lacking a particular property is re-designed to acquire a new diagnostic reasoning system with that particular property, and an existing multi-agent system lacking a particular property is re-designed to acquire a new multi-agent system with that particular property.

Part **III**

# Re-design Models for Compositional Systems

In this part models for re-design of compositional systems are described. A generic model for design is used to construct a model for re-design in the specific domain of compositional systems. A generic design model is described in Chapter 6. Chapter 7 describes a part of the model for re-design of compositional systems that is obtained as a refinement of the generic design model described in Chapter 6. Chapters 8 and 9 extend the presentation of this refinement resulting in a model for re-design of compositional systems. The re-design model is employed in the next Part to construct a *design agent*: an agent capable of re-designing compositional systems.

# 6 A Generic Design Model

A generic model for design is presented in this chapter. This generic model is taken from (Brazier, Langen, Ruttkay and Treur, 1994; for more details refer to Brazier, Langen and Treur, 1999), based on a logical theory of design (Brazier, Langen and Treur, 1996), applied design research (e.g., see (Geelen, Ruttkay and Treur, 1991; Brumsen, Pannekeet and Treur, 1992; Geelen and Kowalczyk, 1992)), and additional applied design research.

This generic model of design has been developed and specified with the DESIRE development method. Both process composition and knowledge composition are specified within the generic design model at an abstract level, thereby leaving room for refinements both of process composition and of knowledge composition. The generic model can be specialised to accommodate various approaches to design, and the generic model can be instantiated for design in a specific domain.

For a more detailed description of this generic model of design, see (Brazier, Langen and Treur, 1999). Applications of the generic design model include an analysis of conflict management in design (Brazier, Langen and Treur, 1997*a*), re-design of knowledge-based systems (Brazier, Langen, Treur and Wijngaards, 1996), elevator configuration design (Brazier, Langen, Treur, Wijngaards and Willems, 1996), design rationale (Brazier, Langen and Treur, 1997*b*), and strategic design knowledge (Brazier, Langen and Treur, 1998*a*).

The description of the generic model of design presented in this chapter does not cover all details, i.e., an abstraction is made of the information types in the generic model to enhance readability. Processes, information exchange and task control are described in detail. In Figure 6.1 the notations are explained which are used to describe information types: abstract 'referring to' and 'meta-description' relationships, and an indication of which information type is intended to be refined with specific domain knowledge. In Table 6.1 the acronyms used throughout this chapter are depicted.

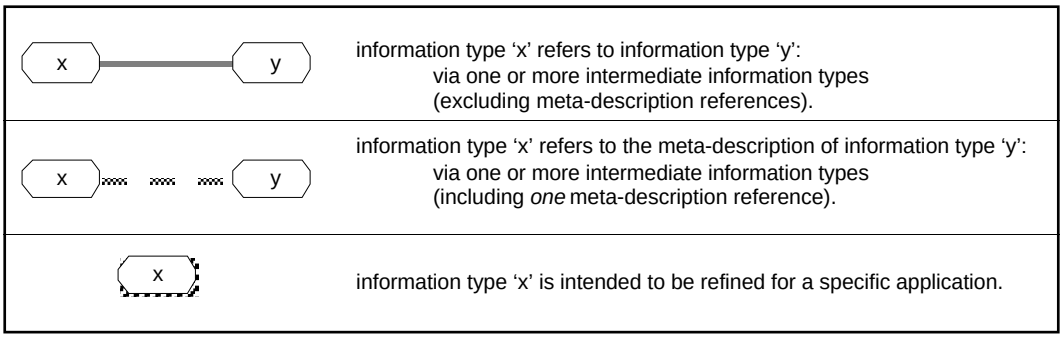


**Figure 6.1** Legend of notations used for abstract description of information types.

Information types described may contain relations, that are generic to design processes. Their actual use (and meaning) can, and will, change depending on the domain of application. Attaching meanings to the pre-defined relations in a specific domain of application aids in investigating and understanding design processes in that particular domain of application.
The compositions of two of the three sub-processes, Design Object Description Manipulation and Requirement Qualification Set Manipulation, are described in this chapter. The composition of the sub-process Design Process Co-ordination is left open. This chapter is structured as follows. In Section 6.1 the composition of the Design process is described, after which the composition of two of the three sub-processes of Design are described: Design Object Description Manipulation

(Section 6.2), and Requirement Qualification Set Manipulation (Section 6.3). A discussion is given in Section 6.4.

| *Acronym* | *Explanation* |
|---|---|
| DOD | Design Object Description |
| RQS | Requirement Qualification Set |
| DPC | Design Process Co-ordination |
| DODM | DOD Manipulation |
| RQSM | RQS Manipulation |

**Table 6.1** Acronyms in use in this chapter.

## 6.1 Composition of design

The process of design is described in three phases: first its process composition, then composition of knowledge structures related to this process, and finally the relations between process composition and knowledge composition.

The description of processes within, and knowledge structures related to DPC, RQS Manipulation, and DOD Manipulation are deferred to respectively Section 6.3, Section 6.2, and Section 6.4.

### 6.1.1 Process composition of design

The process composition of design is described by levels of process abstraction, identification of processes, and composition relation between processes.

The first two levels of process abstraction for Design are shown in Figure 6.2. The processes Design Process Co-ordination, RQS Manipulation, and DOD Manipulation are distinguished within the process Design.

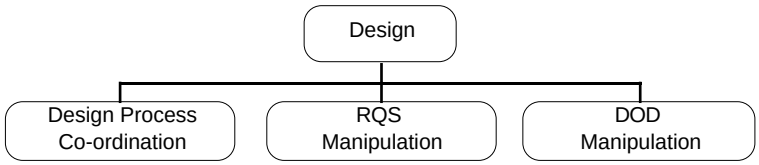


**Figure 6.2** First level of process abstraction within Design.

The process Design Process Co-ordination co-ordinates the design process by issuing information related to overall design strategies on the basis of given design process objectives. The process RQS Manipulation manipulates sets of qualified requirements, on the basis of an overall design strategy, information from DOD Manipulation, and given requirement qualification sets. The process DOD Manipulation manipulates design object descriptions, on the basis of an overall design strategy, information from RQS Manipulation, and given design object descriptions.

Each of the processes depicted in Figure 6.2 can be characterised in terms of their input and output information types, as shown in Table 6.2.

The input and output information types in the interface of the processes described in Table 6.2 is elaborated below:

- The process Design requires, as input, information objectives for the overall design process (design process objectives), a given RQS (RQS) and a given DOD (DOD). The process Design produces an evaluation of the overall design process (design process evaluation), an evaluation of resulting requirement qualification sets (RQS assessment), an evaluation of resulting design object descriptions (DOD assessment), sets of qualified requirements (RQS) and design object descriptions (DOD).

- The process Design Process Co-ordination requires information on objectives for the overall design process (design process objective), and evaluations of the manipulation processes (manipulation process evaluation). The process Design Process Co-ordination produces an evaluation of the overall design process (design process evaluation), and strategies for RQS Manipulation and DOD Manipulation (overall design strategy).
- The process RQS Manipulation requires a strategy (overall design strategy), an evaluation of resulting design object descriptions (DOD assessment), and a given RQS (RQS). The process RQS Manipulation produces an evaluation of the status of its own process (RQSM process evaluation), an evaluation of resulting qualified requirement sets (RQS assessment), and contents of sets of requirement qualifications (RQS).
- The process DOD Manipulation requires an overall design strategy (overall design strategy), information on the requirement qualification set for which a design object description is to be constructed (RQS), and possibly an existing design object description (DOD). The process DOD Manipulation produces an evaluation of the status of its own process (DODM process evaluation), an evaluation of resulting design object descriptions, including information on the satisfaction of design requirements for specific design object descriptions (DOD assessment), and design object descriptions (DOD).

| *process* | *input information type* | *output information type* |
|---|---|---|
| Design | • design process objectives<br>• RQS<br>• DOD | • design process evaluation<br>• RQS assessment<br>• DOD assessment<br>• RQS<br>• DOD |
| Design Process Co-ordination | • design process objective<br>• manipulation process evaluation | • design process evaluation<br>• overall design strategy |
| RQS Manipulation | • overall design strategy<br>• RQS<br>• DOD assessment | • RQSM process evaluation<br>• RQS assessment<br>• RQS |
| DOD Manipulation | • overall design strategy<br>• RQS<br>• DOD | • DODM process evaluation<br>• DOD assessment<br>• DOD |

**Table 6.2** Input and output information types of the process Design and its direct sub-processes.

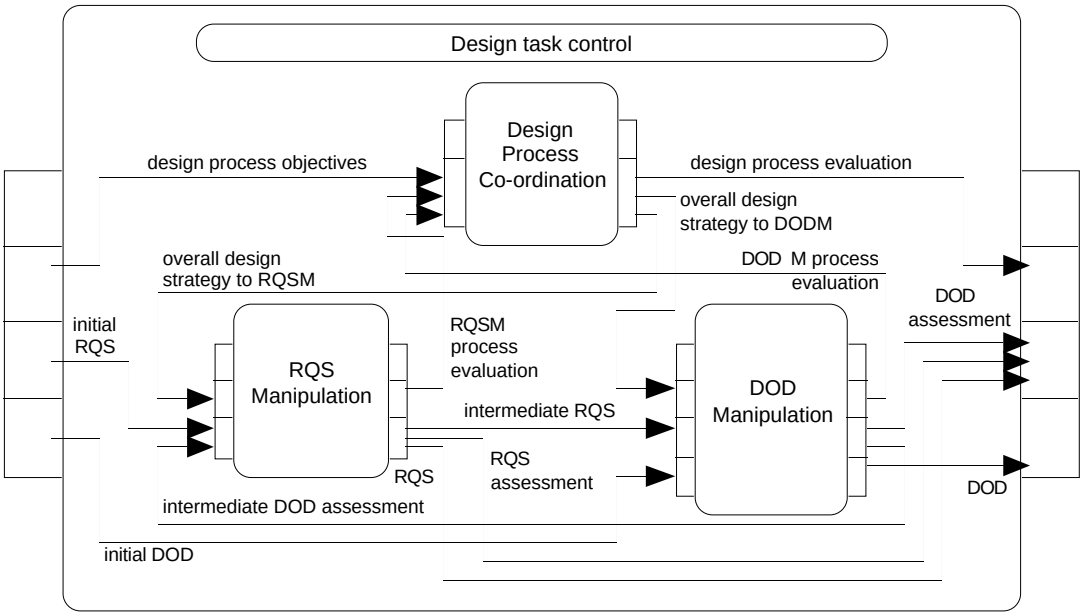


**Figure 6.3** Composition relation between the process of Design and its direct sub-processes.

The static perspective on the composition relation between the process Design and its direct sub-processes is shown in Figure 6.3.

Within the component Design six private links and eight mediating links are defined:

- The mediating link design process objectives transfers design process objectives from the input interface of Design to the input interface of Design Process Co-ordination.
- The mediating link initial RQS transfers RQS from the input interface of Design to the input interface of RQS Manipulation.
- The mediating link initial DOD transfers DOD from the input interface of Design to the input interface of DOD Manipulation.
- The private link overall design strategy to RQSM transfers overall design strategy from the output interface of Design Process Co-ordination to the input interface of RQS Manipulation.
- The private link overall design strategy to DODM transfers overall design strategy from the output interface of Design Process Co-ordination to the input interface of DOD Manipulation.
- The private link RQSM process evaluation transfers RQSM process evaluation from the output interface of RQS Manipulation to the input interface of Design Process Co-ordination.
- The private link DODM process evaluation transfers DODM process evaluation from the output interface of DOD Manipulation to the input interface of Design Process Co-ordination.
- The private link intermediate RQS transfers RQS from the output interface of RQS Manipulation to the input interface of DOD Manipulation.
- The private link intermediate DOD assessment transfers DOD assessment from the output interface of DOD Manipulation to the input interface of RQS Manipulation.
- The mediating link design process evaluation transfers design process evaluation from the output interface of Design Process Co-ordination to the output interface of Design.
- The mediating link RQS assessment transfers RQS assessment from the output interface of RQS Manipulation to the output interface of Design.
- The mediating link RQS transfers RQS from the output interface of RQS Manipulation to the output interface of Design.
- The mediating link DOD assessment transfers DOD assessment from the output interface of DOD Manipulation to the output interface of Design.
- The mediating link DOD transfers DOD from the output interface of DOD Manipulation to the output interface of Design.

The dynamic perspective on the composition relation specifies control over the sub-components of the component Design. Task control within Design specifies: activation of the component Design, possible terminations of DPC, possible terminations of RQS Manipulation and possible terminations of DOD Manipulation.

- Upon activation of the component Design the component DPC is activated and the information links design process objectives, initial RQS, and initial DOD are made up-to-date.
- Upon termination of the component DPC, and *successful* determination of an overall design strategy, both the components RQS Manipulation and DOD manipulation are activated. The information links overall design strategy to RQSM and overall design strategy to DODM are made up-to-date.
- The components RQS Manipulation and DOD Manipulation are both active and react to the given overall design strategy. If a component has become idle, it has finished its own task (if any) in accordance with the given strategy. When *both* components are idle, then the component DPC is activated, and the information links RQSM process evaluation, intermediate RQS, DODM process evaluation, and intermediate DOD assessment are made up-to-date.
- Upon termination of component DPC, and *failure* to determine an overall design strategy, the design process is terminated, and the information links design process evaluation, RQS assessment, RQS, DOD assessment, and DOD are made up-to-date.

The task control described above leaves open whether RQS Manipulation and DOD Manipulation operate in parallel or sequentially: the overall design strategy may indicate which sub-process is to be activated. This is one of the issues which a strategy needs to address.

### 6.1.2 Knowledge composition of design

The information types used in the interfaces of the component Design and its direct sub-components are briefly described in this section. The contents of the main information types are relations, which are described in the concise, textual notation.

**DOD.** The information type DOD is based on meta-descriptions of two other information types: basic design object information (e.g., facts representing a design object), and derivable design object information (e.g., properties of a design object). Each DOD has a name, and associated basic and derivable design object information. In Figure 6.4 this view on the information type DOD is depicted. The information types basic design object information and derivable design object information are intended to be instantiated for a specific domain.

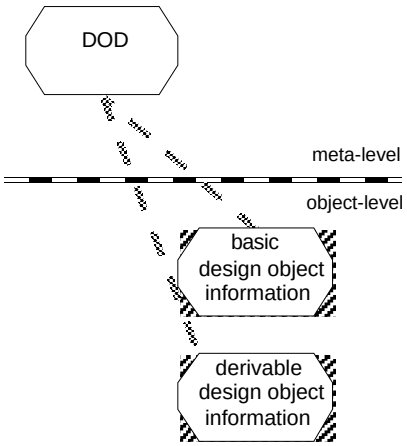


**Figure 6.4** Partial view on information type DOD.

The generic relation defined in the information types basic design object information and derivable design object information is:

```
relations
    has_value:                  domain_object *
                                attribute *
                                value;
```

All three sorts in the above relation are intended to be instantiated. In addition, domain specific relations can be added. Specific relations for basic design object information can be added, and the same holds for the information type derivable design object information.

The generic relation defined in the information type DOD is:

```
relations
    includes_design_object_information:   DOD_name *
                                          design_object_info_element *
                                          sign;
```

In the relation includes_design_object_information the name of a DOD is related to a specific domain dependent contents, with a sign (which denotes explicit presence, or explicit absence). The sort design object info element contains the meta-descriptions of the information types basic design object information and derivable design object information and is used to describe the contents of a DOD.

**RQS.** The term ‘design requirement’ is used for both unqualified requirements and qualified requirements. The information type RQS is based on meta-descriptions of two other information

types: basic design requirement information (e.g., facts representing design requirements), and derivable design requirement information (e.g., properties of design requirements, refined design requirements). Each RQS has a name, and associated basic and derivable design requirement information. In Figure 6.5 this view on the information type RQS is depicted. The information types basic design requirement information and derivable design requirement information are both intended to be instantiated.

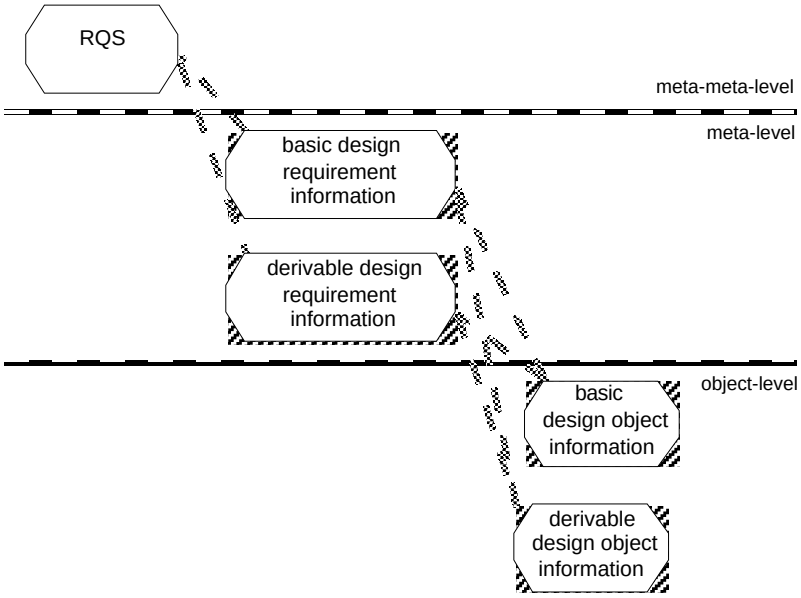


**Figure 6.5** Partial view on information type RQS.

Generic relations available in the information types basic design requirement information and derivable design requirement information are:

**relations**

| | |
|---|---|
| is_requirement: | requirement_name * |
| | design_object_info_element_expression; |
| is_qualified_requirement: | qualified_requirement_name * |
| | qualification * |
| | requirement_name_tuple; |

The sort design object info element expression contains the meta-descriptions of basic and derivable design object information, on which logical operators are defined (to be able to formulate expressions).

The generic relations defined in the information type RQS are:

**relations**

| | |
|---|---|
| includes_qualified_requirement_information: | RQS_name * |
| | design_requirement_info_element; |

The sort design requirement info element contains meta-descriptions of the information types basic design requirement information and derivable design requirement information.

**DOD assessment.** The information type DOD assessment is based on five information types: DOD properties, DOD evaluation, DOD relations, RQS evaluation, and DOD appreciation, as shown in Figure 6.6. The information type DOD evaluation consists of the information types DOD basis evaluation and DOD fulfillment. The information type RQS evaluation consists of the information types RQS basis evaluation and RQS realisation evaluation. The information type DOD properties contains relations on properties of individual design object descriptions. The information type DOD basis evaluation contains relations between a DOD and satisfaction of design requirements. The information type DOD fulfillment evaluation contains relations on the fulfillment of sets of qualified requirements by design object descriptions. The information type DOD relations

contains relations on comparisons between design object descriptions. The information type RQS basis evaluation contains relations on evaluations between design requirements. The information type RQS realisation evaluation contains relations on the feasibility of sets of qualified requirements. The information type DOD appreciation contains relations on appreciation of parts of design object descriptions.

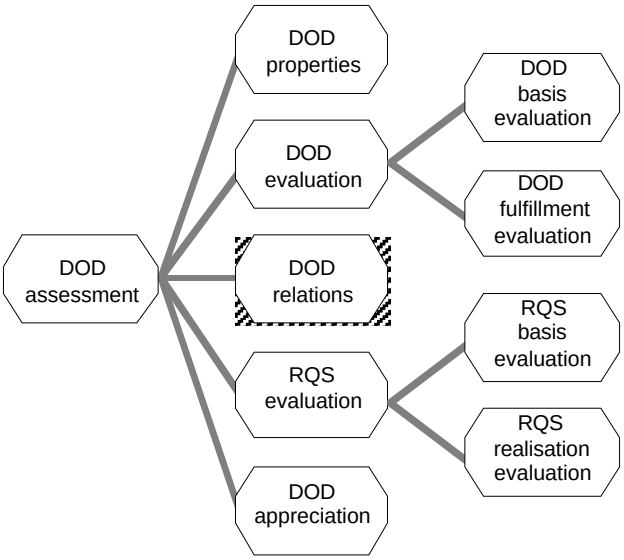


**Figure 6.6** Partial view on information type DOD assessment.

Generic relations defined in the information type DOD properties are:

**relations**
is_consistent,
is_deductively_consistent,
is_deductively_maximum,
is_deductively_minimum: DOD_name;

These relations express consistency of design object descriptions and additional information acquired by deduction, and information on the extent to which deduction has been performed.

Generic relations defined in the information type DOD basis evaluation are:

**relations**
satisfies,
violates: DOD_name * requirement_name;
supports,
undermines: DOD_name * qualified_requirement_name;

These relations express for a given design object description whether a requirement is satisfied or violated (but not both), and whether a qualified requirement is supported or undermined (but not both).

Generic relations defined in the information type DOD fulfillment evaluation are:

**relations**
fulfills,
falls_short_of: DOD_name * RQS_name;

These relations indicate whether a design object description fulfills a set of qualified requirements, or falls short of fulfilling the RQS in its entirety (but not both).

Generic relations defined in the information type DOD relations are:

**relations**
is_deductive_refinement_of,
is_deductive_closure_of,
is_deductive_core_of,

is_consistent_with,
is_deductively_consistent_with,
differs_essentially_from,
extends: DOD_name * DOD_name;

These relations express relations between two design object descriptions. Whether a DOD is consistent with, in conflict with, a refinement of, or an extension of another DOD is information that expresses how specific design object descriptions are related. Additional relations can be added to this information type.

Generic relations defined in the information type DOD appreciation are:

**relations**
has_appreciation: DOD_name *
design_object_info_element *
appreciation_value;

This relation describes which parts of a design object description are criticised to a certain extent: e.g., alternative solutions, or a solution which does not conform to current design requirements.

Generic relations defined in the information types RQS basis evaluation and RQS realisation evaluation are described in description of the information type RQS assessment.

**RQS assessment.** The information type RQS assessment is based on four information types: RQS properties, RQS relations, RQS appreciation, and RQS evaluation, as shown in Figure 6.7. The information type RQS evaluation consists of the information types RQS basis evaluation and RQS fulfillment evaluation. The information type RQS basis evaluation consists of the information type requirement evaluation and qualified requirement evaluation. The information type RQS properties contains relations on properties of individual sets of qualified requirements. The information type RQS basis evaluation contains relations on evaluations between design requirements. The information type RQS fulfillment evaluation contains relations on the feasibility of sets of qualified requirements. The information type RQS relations contains relations on comparisons between sets of qualified requirements. The information type RQS appreciation contains relations criticising parts of sets of qualified requirements.

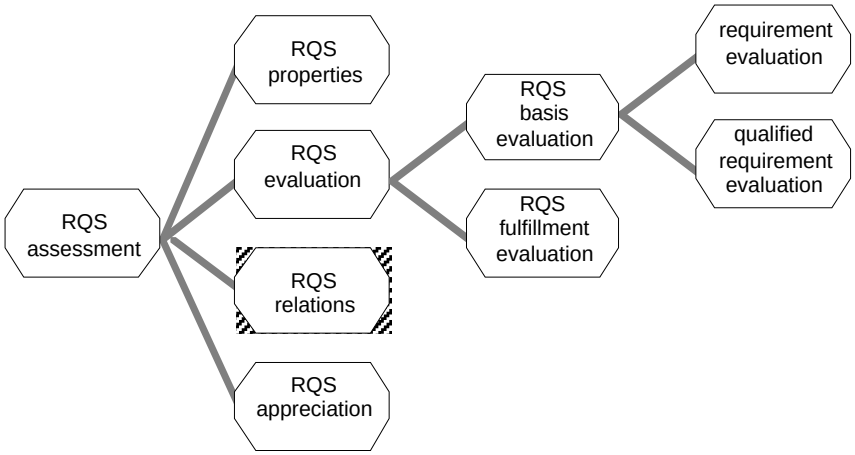


**Figure 6.7** Partial view on information type RQS assessment.

Generic relations defined in the information type RQS properties are:

**relations**
is_consistent,
is_deductively_consistent,
is_deductively_maximum,
is_deductively_minimum: RQS_name;

These relations express consistency of sets of qualified requirements and additional information acquired by deduction, and information on the extent to which deduction has been performed.

Generic relations in the information type requirement evaluation are:

**relations**

| | |
|---|---|
| is_missing: | requirement_expression; |
| is_satisfiable, | |
| is_unsatisfiable, | |
| is_precise, | |
| is_imprecise: | requirement_name; |
| is_compatible_with, | |
| is_incompatible_with: | requirement_name * |
| | requirement_name; |

The relation is missing expresses that a particular expression is missing from the current set of requirements. The relations is satisfiable and is unsatisfiable express whether or not it is possible to satisfy a requirement on the basis of available knowledge. The relations is precise and is imprecise express whether or not a requirement is too imprecise to be handled. The relations is compatible with and is incompatible with express whether or not a requirement is in a conflict with another requirement.

Generic relations in the information type qualified requirement evaluation are:

**relations**

| | |
|---|---|
| is_supportable, | |
| is_unsupportable, | |
| is_unambiguous, | |
| is_ambiguous: | qualified_requirement_name; |
| agrees_with, | |
| disagrees_with: | qualified_requirement_name * |
| | qualified_requirement_name; |

The relations is supportable and is unsupportable express whether or not it is possible to support a qualified requirement on the basis of available knowledge. The relations is unambiguous and is ambiguous express whether or not a qualified requirement can be refined into more specific qualified requirements. The relations agrees with and disagrees with express conflicts between qualified requirements.

Generic relations defined in the information type RQS fulfillment evaluation are:

**relations**

| | |
|---|---|
| is_realisable, | |
| is_unrealisable, | |
| is_harmonious, | |
| is_unharmonious, | |
| is_precise, | |
| is_imprecise, | |
| is_complete, | |
| is_incomplete, | |
| is_unambiguous, | |
| is_ambiguous: | RQS_name; |

The relations is realisable and is unrealisable express whether or not a set of qualified requirements can be realised by constructing a design object description fulfilling the RQS. The relations is harmonious and is unharmonious express whether or not a RQS is internally consistent. The relations is precise and is imprecise express whether or not a set of qualified requirements is precise enough, or contains design requirements which are too ‘vague’ to be handled. The relations is complete and is incomplete express whether or not a set of qualified requirements is considered to be complete. The relations is unambiguous and is ambiguous express whether or not a set of qualified requirements is internally unambiguous.

Generic relations defined in the information type RQS relations are:

**relations**

| |
|---|
| is_deductive_refinement_of, |
| is_deductive_closure_of, |
| is_deductive_core_of, |
| is_consistent_with, |

is_deductively_consistent_with,
differs_essentially_from,
extends: RQS_name * RQS_name;

These relations express relations between two sets of qualified requirements. Whether a RQS is consistent with, in conflict with, a refinement of, or an extension of another RQS is information that expresses how specific sets of qualified requirements are related. Additional relations can be added to this information type.

Generic relations defined in the information type RQS appreciation are:

**relations**
has_appreciation: RQS_name *
design_requirement_info_element *
appreciation_value;

This relation describes which parts of a set of qualified requirements are criticised to a certain extent: e.g., alternative solutions.

The information types manipulation process evaluation (including RQSM process evaluation and DODM process evaluation), design process objectives, design process evaluation, and overall design strategy are described in Appendix A.1.

### 6.1.3 Relation between process composition and knowledge composition of design

The information types previously identified in the process identification of the processes Design, Design Process Co-ordination, RQS Manipulation, and DOD Manipulation have been described in the knowledge composition and are related to these processes. Knowledge bases have not been specified.

## 6.2 Composition of design object description manipulation

The composition of DOD Manipulation is described in three phases: first the process composition is described, then the knowledge composition of Design (Section 6.1.2), and finally the relation between process composition and knowledge composition is described.

### 6.2.1 Process composition of DODM

The process composition of DOD Manipulation is described by levels of process abstraction, identification of processes, and composition relation between processes.

The first level of process abstraction for DOD Manipulation is shown in Figure 6.8. The processes DOD Modification, DODM History Maintenance, deductive DOD refinement, and current DOD maintenance are distinguished within the process DOD Manipulation.

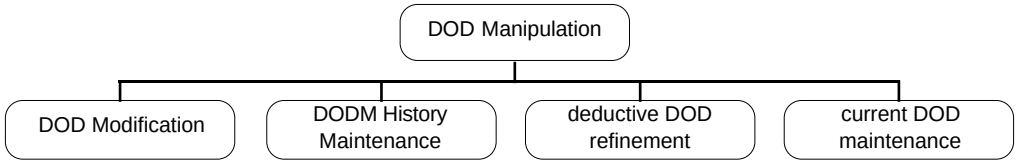


**Figure 6.8** Processes at different abstraction levels in DOD Manipulation.

The process DOD Modification plays an important role within DOD Manipulation: on the basis of the overall design strategy, a given DOD, given design requirements, and information from DODM history maintenance, design object descriptions are considered and modified. The process DODM History Maintenance stores and retrieves information related the overall DODM process. The process deductive DOD refinement deduces properties of design object descriptions, and the component current DOD maintenance contains the contents of the current DOD. The process DOD Modification determines which actions are taken within the DOD Manipulation process, e.g., when

to replace the current DOD, when to start deductively refining the current DOD, etc. The co-ordination of the sub-processes of DOD Manipulation is planned *within* DOD Modification.

Each of the processes depicted in Figure 6.8 can be characterised in terms of their input and output information types, as shown in Table 6.3.

| *process* | *input information type* | *output information type* |
|---|---|---|
| DOD Modification | • DOD modification state history search results<br>• overall design strategy<br>• DOD assessment history search results<br>• RQS history search results<br>• DOD history search results<br>• current DOD replacement results<br>• current DOD contents | • DOD modification progress<br>• current manipulation action<br>• DOD modification state history queries<br>• DOD assessment<br>• DOD assessment history queries<br>• RQS history queries<br>• DOD refinement focus<br>• current DOD focus<br>• current DOD modification<br>• DOD history queries<br>• current DOD replacement request |
| DODM History Maintenance | • DOD modification progress<br>• DOD modification state history queries<br>• overall design strategy<br>• DOD assessment<br>• DOD assessment history queries<br>• RQS<br>• RQS history queries<br>• DOD<br>• DOD history queries<br>• current DOD replacement request<br>• current DOD contents | • DOD modification state history search results<br>• DOD assessment history search results<br>• RQS history search results<br>• DOD history search results<br>• current DOD replacement results<br>• new current DOD contents |
| deductive DOD refinement | • basic design object information | • design object information |
| current DOD maintenance | • design object information | • design object information |

**Table 6.3** Input and output information types of the DOD Manipulation process and its direct sub-processes.

The processes described in Table 6.3 require, as input, the input information types, and produce, as output, the output information types. A more extended description of the interfaces of these processes can be found in Appendix A.2.

The static perspective on the composition relation between the process DOD Manipulation and its sub-processes, is shown in Figure 6.9. The information links shown in Figure 6.9 have names which reflect the information types transferred from the origin of the information link to the destination of the information link. These information links are described in Appendix A.2.2.

The dynamic perspective on the composition relation includes specification of the control over the sub-components of the component DOD Manipulation. The task control within DOD Manipulation distinguishes a number of phases. Upon activation of DOD Manipulation, DOD Modification is activated. DOD Modification can choose between continuation of the previous manipulation process, or initiation of a new manipulation process. DOD Modification may produce queries on history information and requirement qualification sets, and obtain results from these queries, and DOD Modification can indicate that a DOD has to become the current DOD (i.e., placed in the component current DOD maintenance). When a current (possibly empty) DOD is available, refinements can be deduced, foci on the current DOD can be made, modifications can be applied, and information (progress of the modification process and current DOD) can be stored in the history. When DOD Modification establishes that the manipulation of design object descriptions has finished, specific information can be made available as output of DOD Manipulation (e.g., via retrieval of information in the histories) and DOD Manipulation then terminates itself.

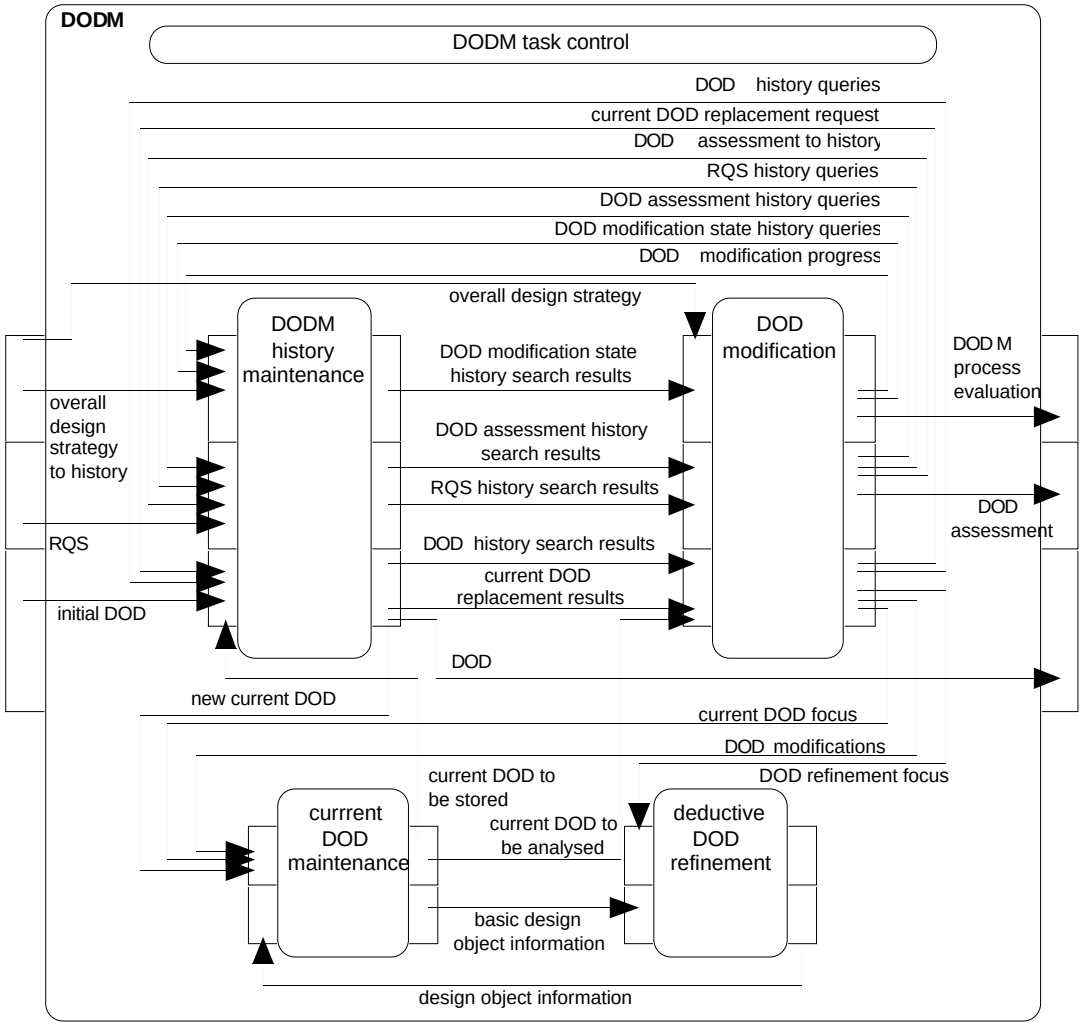


**Figure 6.9** Information exchange in DOD Manipulation.

### 6.2.2 Knowledge composition of DODM

Information types and knowledge bases related to the sub-components of the component DOD Manipulation are briefly described in this section and in Appendix A.2. In this section the information types related to design object information and manipulation of the current DOD are described. The contents of the main information types are relations which are described in the concise, textual notation.

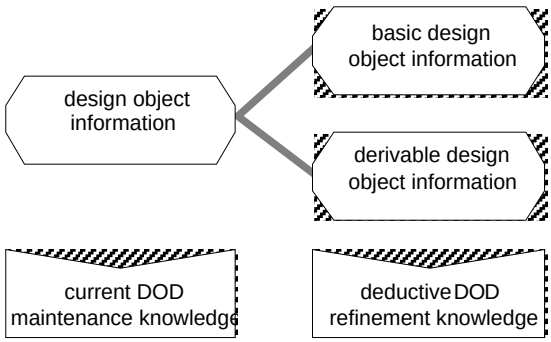


**Figure 6.10** Information types related to design object information and two knowledge bases.

**Design object information & knowledge bases.** The information type design object information consists of two information types: basic design object information and derivable design object information as shown in Figure 6.10. In addition, two knowledge bases are depicted which correspond to knowledge needed for the processes current DOD maintenance and deductive DOD refinement.

The information types basic design requirement information and derivable design requirement information are explained in Section 6.1. The knowledge bases current DOD maintenance and deductive DOD refinement contain knowledge on how to focus on parts of the contents of a DOD, and knowledge on deductive refinement of the contents of a DOD, respectively.

**Information types related to manipulation of the current DOD.** The information types shown in Figure 6.11 are all related to the manipulation of a current DOD. The information types current DOD focus, current DOD modification, DOD refinement focus, current DOD replacement request, current DOD replacement results, current DOD contents, and new current DOD contents are directly related to manipulation: focussing on part of the current DOD, modifying the current DOD, directing the deductive refinement of the current DOD, replacing the current DOD, results on the success of replacing the current DOD, the contents of the current DOD, and the contents of the new DOD. The information type current DOD modification consists of the three information types possible DOD modification, rejected DOD modification, and selected DOD modification.

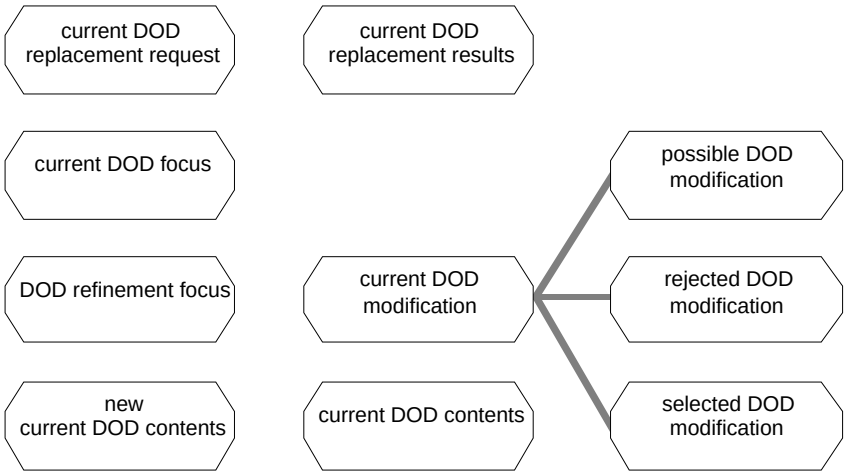


**Figure 6.11** Information types related to manipulating a current DOD.

The generic relation defined in the information type current DOD focus is:

```
relations
    is_part_of_current_DOD_focus:      design_object_information_atom *
                                       sign;
```

This relation describes which design requirements are part of the current DOD focus, and which design requirements are *not* part of that focus.

The generic relation defined in the information type current DOD replacement request is:

```
relations
    is_new_current_DOD:                DOD_name;
```

This relation describes which DOD in the DODM history, is to be used as the current DOD.

The generic relations defined in the information type current DOD replacement results is:

```
relations
    is_current_DOD,
    is_not_current_DOD:                DOD_name;
```

The relation is current DOD expresses that the contents of a design object description is used as the current DOD. The relation is not current DOD expresses that the design object description is not used as the current DOD, e.g. because this design object description is unknown to the history maintenance process.

The generic relation defined in the information type current DOD contents is:

**relations**
holds: design_object_information_atom * sign;

This relation describes the contents of the current DOD.

The generic relation defined in the information type new current DOD contents is:

**relations**
is_part_of_new_current_DOD: design_object_information_atom * sign;

This relation describes the new contents of the current DOD.

The generic relation defined in the information type DOD refinement focus is:

**relations**
is_part_of_DOD_refinement_focus: design_object_information_atom * sign;

This relation describes whether a (derivable) design object element is in focus, or not, for deductive refinement.

Generic relations defined in the information types possible DOD modification, rejected DOD modification, and selected DOD modification are:

**relations**
is_possible_design_object_element_for_addition,
is_possible_design_object_element_for_deletion,
is_rejected_design_object_element_for_addition,
is_rejected_design_object_element_for_deletion,
is_selected_design_object_element_for_addition,
is_selected_design_object_element_for_deletion: design_object_information_atom * sign;

These relations describe whether a design object element: may be added or deleted, may *not* be added or deleted, and is selected to be added or deleted.

### 6.2.3 Relation between process composition and knowledge composition of DODM

The information types in the interfaces of the sub-components of DOD Manipulation, as described in section 6.2.1, are defined in section 6.2.2. These information types are related to the sub-components of DOD Manipulation according to the description of the interfaces of these sub-components. The two knowledge-bases are related to the sub-processes current DOD maintenance and deductive DOD refinement.

## 6.3 Composition of requirement qualification set manipulation

The composition of RQS Manipulation is described in three phases: first the process composition is described, then the knowledge composition, and finally the relation between process composition and knowledge composition is described.

### 6.3.1 Process composition of RQSM

The process composition of RQS Manipulation is described by levels of process abstraction, identification of processes, and composition relation between processes.

The first level of process abstraction for RQS Manipulation is shown in Figure 6.12. The processes RQS Modification, RQSM History Maintenance, deductive RQS refinement, and current RQS maintenance are distinguished within the process RQS Manipulation.

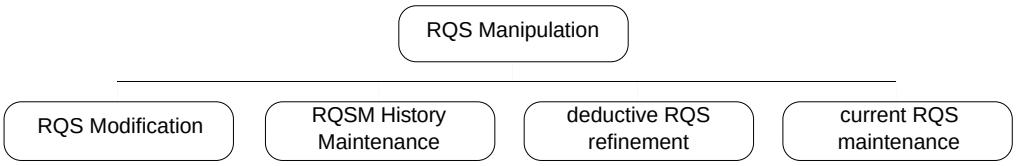


**Figure 6.12** Processes at different abstraction levels in RQS Manipulation.

The process RQS Modification plays an important role within RQS Manipulation: on the basis of the overall design strategy, given RQS and information from DODM, sets of qualified requirements are considered and modified. The process RQSM History Maintenance stores and retrieves information related the overall RQSM process. The process deductive RQS refinement deduces properties of design requirements, and the component current RQS maintenance contains the contents of the current RQS. The process RQS Modification determines which actions are to be taken within the RQS Manipulation process, e.g., when to replace the current RQS, when to start deductively refining the current RQS, etc. The co-ordination of the sub-processes of RQS Manipulation is planned *within* RQS Modification.

Each of the processes depicted in Figure 6.12 can be characterised in terms of their input and output information types, as shown in Table 6.4.

| *process* | *input information type* | *output information type* |
|---|---|---|
| RQS Modification | • RQS modification state history search results<br>• overall design strategy<br>• RQS assessment history search results<br>• DOD assessment history search results<br>• RQS history search results<br>• current RQS replacement results<br>• current RQS contents | • RQS modification progress<br>• current manipulation action<br>• RQS modification state history queries<br>• RQS assessment<br>• RQS assessment history queries<br>• DOD assessment history queries<br>• RQS refinement focus<br>• current RQS focus<br>• current RQS modification<br>• RQS history queries<br>• current RQS replacement request |
| RQSM History Maintenance | • RQS modification progress<br>• RQS modification state history queries<br>• overall design strategy<br>• RQS assessment<br>• RQS assessment history queries<br>• DOD assessment<br>• DOD assessment history queries<br>• RQS<br>• RQS history queries<br>• current RQS replacement request<br>• current RQS contents | • RQS modification state history search results<br>• RQS assessment history search results<br>• DOD assessment history search results<br>• RQS history search results<br>• current RQS replacement results<br>• new current RQS contents |
| deductive RQS refinement | • basic design requirement information | • design requirement information |
| current RQS maintenance | • design requirement information | • design requirement information |

**Table 6.4** Input and output information types of the RQS Manipulation process and its direct sub-processes.

The processes described in Table 6.4 require, as input, the input information types, and produce, as output, the output information types. A more extended description of the interfaces of these processes can be found in Appendix A.3.

The static perspective on the composition relation between the process RQS Manipulation and its sub-processes, is shown in Figure 6.13.

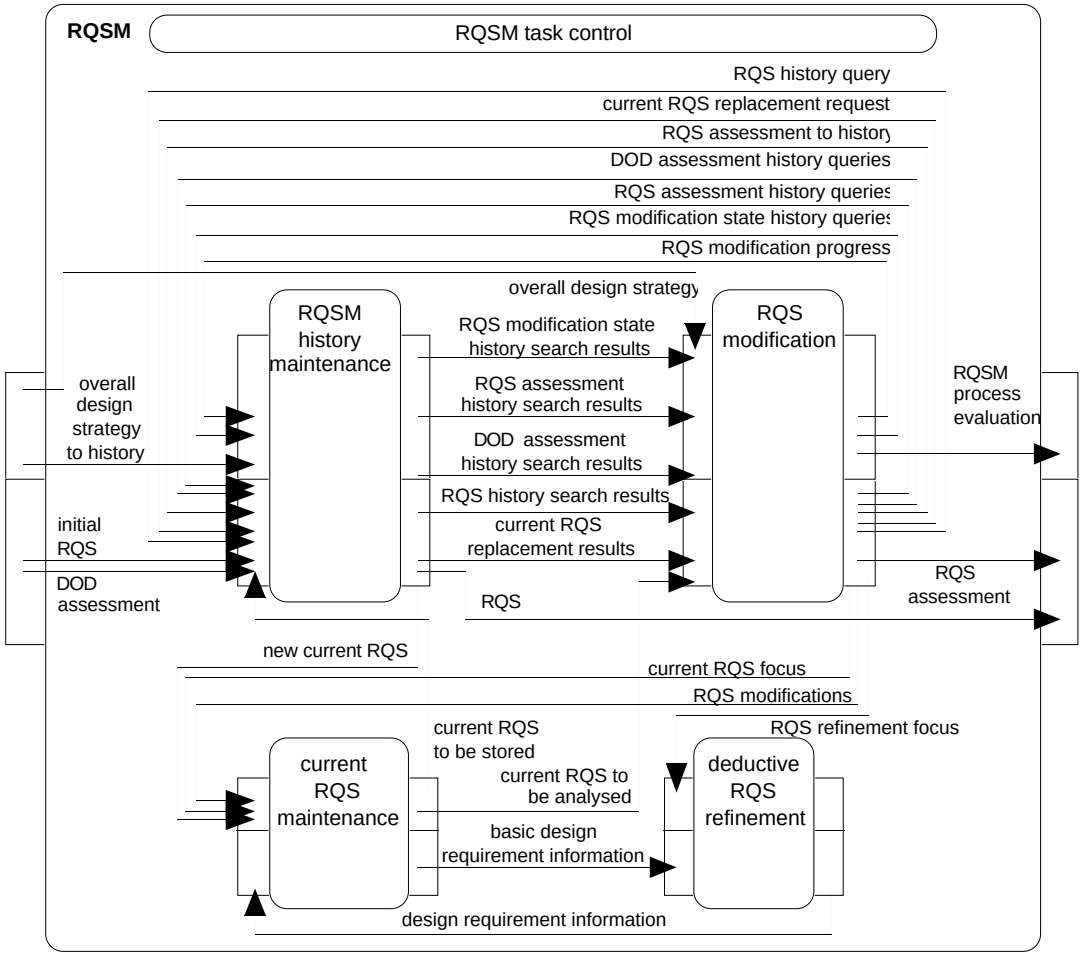


**Figure 6.13** Information exchange in RQS Manipulation.

The information links shown in Figure 6.13 have names which reflect the information types transferred from the origin of the information link to the destination of the information link. These information links are described in Appendix A.3.2.

The dynamic perspective on the composition relation includes specification of the control over the sub-components of the component RQS Manipulation. The task control within RQS Manipulation distinguishes a number of phases. Upon activation of RQS Manipulation, RQS Modification is activated. Then RQS Modification can choose between continuation of the previous manipulation process, or initiation of a new manipulation process. RQS Modification may produce queries on history information, and obtain results from these queries, and RQS Modification can indicate that a RQS has to become the current RQS (i.e., placed in the component current RQS maintenance). When a current (possibly empty) RQS is available, refinements can be deduced, foci on the current RQS can be made, modifications can be applied, and information (progress of the modification process and current RQS) can be stored in the history. When RQS Modification establishes that the manipulation of requirement qualification sets has finished, specific information can be made available as output of RQS Manipulation (e.g., via retrieval of information in the histories) and RQS Manipulation then terminates itself.

### 6.3.2 Knowledge composition of RQSM

Information types and knowledge bases related to the sub-components of the component RQS Manipulation are briefly described in this section and in Appendix A.3. In this section the

information types related to design requirement information and manipulation of the current RQS are described. The contents of the main information types are relations which are described in the concise, textual notation.

**Design requirement information & knowledge bases.** The information type design requirement information consists of two information types: basic design requirement information and derivable design requirement information as shown in Figure 6.14. In addition, two knowledge bases are depicted which correspond to knowledge needed for the processes current RQS maintenance and deductive RQS refinement.

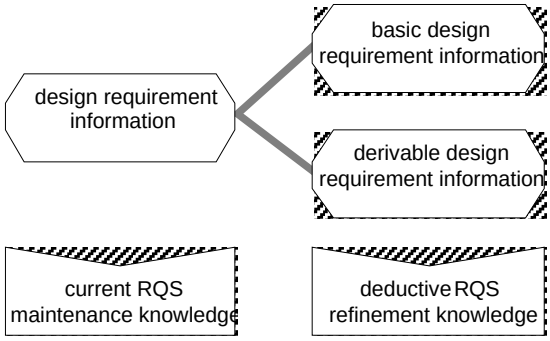


**Figure 6.14** Information types related to design requirement information and two knowledge bases.

The information types basic design requirement information and derivable design requirement information are explained in Section 6.1. The knowledge bases current RQS maintenance and deductive RQS refinement contain knowledge on how to focus on parts of the contents of a RQS, and knowledge on deductive refinement of the contents of a RQS, respectively.

**Information types related to manipulation of the current RQS.** The information types shown in Figure 6.15 are all related to the manipulation of a current RQS. The information types current RQS focus, current RQS modification, RQS refinement focus, current RQS replacement request, current RQS replacement results, current DOD contents, and new current RQS contents are directly related to manipulation: focussing on part of the current RQS, modifying the current RQS, directing the deductive refinement of the current RQS, replacing the current RQS, results on the success of replacing the current RQS, contents of the current RQS, and the contents of the new RQS. The information type current RQS modification consists of the three information types possible RQS modification, rejected RQS modification, and selected RQS modification.

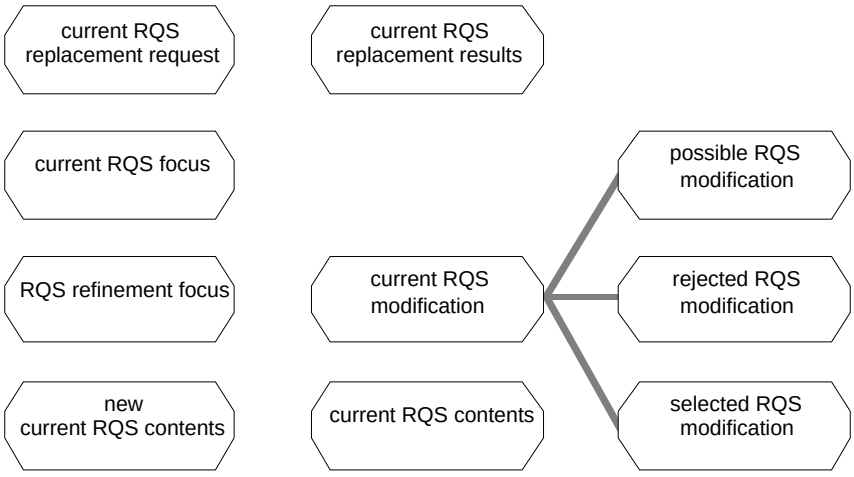


**Figure 6.15** Information types related to manipulating a current RQS.

The generic relation defined in the information type current RQS focus is:

**relations**
is_part_of_current_RQS_focus: design_requirement_information_atom * sign;

This relation describes which design requirements are part of the current RQS focus, and which design requirements are *not* part of that focus.

The generic relation defined in the information type current RQS replacement request is:

**relations**
is_new_current_RQS: RQS_name;

This relation describes which RQS in the RQSM history, is to be used as the current RQS.

The generic relations defined in the information type current RQS replacement results is:

**relations**
is_current_RQS,
is_not_current_RQS: RQS_name;

The relation is current RQS expresses that the contents of a set of qualified requirements is used as the current RQS. The relation is not current RQS expresses that the set of qualified requirements is not used as the current RQS, e.g. because this set of qualified requirements is unknown to the history maintenance process.

The generic relation defined in the information type current RQS contents is:

**relations**
holds: design_requirement_information_atom * sign;

This relation describes the contents of the current RQS.

The generic relation defined in the information type new current RQS contents is:

**relations**
is_part_of_new_current_RQS: design_requirement_information_atom * sign;

This relation describes the new contents of the current RQS.

The generic relation defined in the information type RQS refinement focus is:

**relations**
is_part_of_RQS_refinement_focus: design_requirement_information_atom * sign;

This relation describes whether a (derivable) design requirement is in focus, or not, for deductive refinement.

Generic relations defined in the information types possible RQS modification, rejected RQS modification, and selected RQS modification are:

**relations**
is_possible_design_requirement_for_addition,
is_possible_design_requirement_for_deletion,
is_rejected_design_requirement_for_addition,
is_rejected_design_requirement_for_deletion,
is_selected_design_requirement_for_addition,
is_selected_design_requirement_for_deletion: design_requirement_information_atom * sign;

These relations describe whether a design requirement: may be added or deleted, may *not* be added or deleted, and is selected to be added or deleted.

### 6.3.3 Relation between process composition and knowledge composition of RQSM

The information types in the interfaces of the sub-components of RQS Manipulation, as described in section 6.3.1, are defined in section 6.3.2. These information types are related to the sub-components of RQS Manipulation according to the description of the interfaces of these

sub-components. The two knowledge-bases are related to the sub-processes current RQS maintenance and deductive RQS refinement.

## 6.4 Discussion

In this chapter a generic model of design has been described. The process design has been shown to consist of the processes design process co-ordination, requirement qualification set (RQS) manipulation, and design object description (DOD) manipulation. The process composition and knowledge composition have been described for the processes design, RQS manipulation and DOD manipulation. More detailed descriptions of the process and knowledge composition are provided in Appendix A.

This description of the generic design model abstracts from a number of details. For a more detailed description of this generic model, see (Brazier, Langen and Treur, 1999).

The generic design model contains separate processes for reasoning about design strategies, sets of qualified requirements, and design object descriptions. Each of these processes can be specialised (and related knowledge structures can be instantiated) for a design process in a specific domain. It is not necessary to make use of all of the processes and knowledge structures, e.g., consider the process design process co-ordination: if DPC is not necessary in a specific design process, it can be given a trivial content or be left out of that specific design model entirely. In another situation, the process RQS manipulation may not be necessary and may be given trivial content.

This generic model can be employed for 'model-driven knowledge engineering': each of the generic processes and knowledge structures *could* play a role in a specific design process.

A number of desired properties of a design model, to be used as a basis for the research presented in this thesis, have been identified in Section 2.1. These desired properties concern:

- the explicit distinction between manipulation of design object descriptions, manipulation of sets of qualified requirements, and co-ordination of the design process;
- explicit representation and manipulation of design process objectives;
- explicit representation and manipulation of (sets of) qualified requirements;
- explicit representation and manipulation of design object descriptions.

These desired properties are satisfied by the generic design model described in this chapter: the processes DPC, RQS Manipulation, and DOD Manipulation are distinguished as separate processes which manipulate specific information, which is defined by specific information types which contain representations of design process objectives, (sets of) qualified requirements, and design object descriptions.

This generic design model is the basis for a model of *re-design*, as described in the next chapter.

# 7 A Re-design Model for Compositional Systems

The generic model for design described in the previous chapter can be refined by specialisation and instantiation. This chapter and the next two chapters describe a refinement of the generic model of design for the domain of re-design of compositional systems. Five desiderata are directly related to a model for re-design of compositional systems:

- dr1, representation of a knowledge-intensive system,
- dr2, representation of qualified requirements on compositional systems,
- dr3, model of the manipulation of the structure of compositional systems,
- dr4, model of the manipulation of requirements on compositional systems, and
- dr5, knowledge on the co-ordination of the re-design process.

The realisation of these desiderata results in a refinement step as depicted in Figure 7.1: the generic design model is refined into a model for re-design of compositional systems.

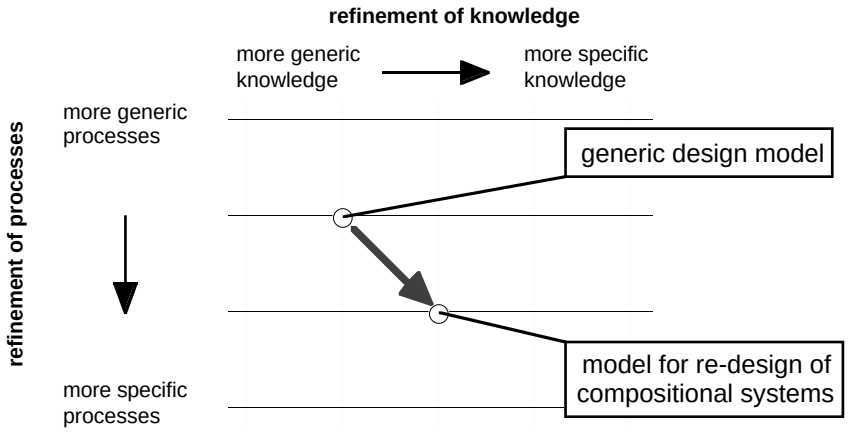


**Figure 7.1** Intended target of refinement of generic design model.

The refinement step depicted in Figure 7.1 is split up into three smaller refinement steps, as illustrated in Figure 7.2. The first step is the refinement of the information types related to the interface of the design process and the refinement of the process design process co-ordination (in this chapter). The second refinement step is the refinement of the process RQS manipulation (Chapter 8). The third refinement step is the refinement of the process DOD manipulation (Chapter 9).

During each refinement step depicted in Figure 7.2, refinement is realised by:

- specialisation of the process composition: more specific processes are identified within these three processes, with their interface information types and composition;
- instantiation of the knowledge composition: specific knowledge structures are identified as are their composition;
- refinement of the relation between the more specialised process composition and instantiated knowledge composition.

The description of the model of re-design presented in this chapter (and the next two chapters) does not cover all details, i.e., an abstraction is made of the information types in this model to enhance readability while constraining the number of information types explained. Similarly, not all processes of re-design are explained in detail: information exchange and task control are described for a given level of abstraction.

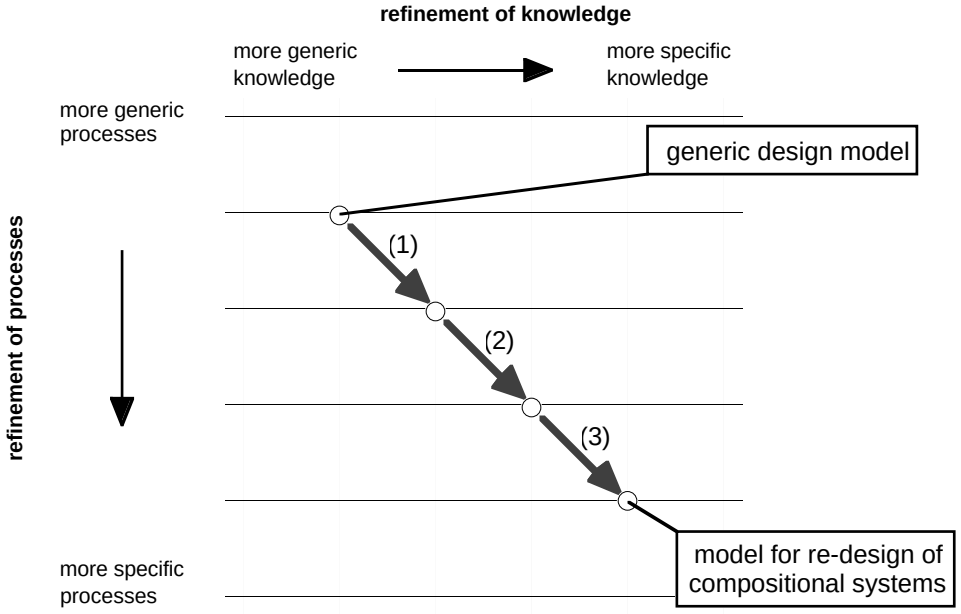


**Figure 7.2** Stepwise approach to refine the generic design model into a model for re-design of compositional systems.

In Figure 6.1 the notations used to describe information types are explained: abstract 'referring to' and 'meta-description' relationships, and an indication that a given information type is defined in the generic model of design (see Chapter 6). Acronyms as listed in Table 6.1 are also employed in this chapter.

In this chapter the realisation of desiderata dr1, dr2 and dr5 is described. In Section 7.1 some of the information types related to the interface of the design process are instantiated. The process design process co-ordination is refined in Section 7.2. Section 7.3 briefly discusses the results of the refinement of the generic design model.

## 7.1 Instantiation of top-level interface information types

The generic information types related to the interface of the process Design have been described in Section 6.1. A number of these information types are intended to be instantiated by information related to the application domain for the design model.

Information types defining sorts which represent names (e.g., RQS names, DOD names, name of a qualified requirement) are not shown in detail in Chapter 6, but need to be instantiated. The sort strings is a sub-sort of all sorts which represent names, allowing for the formulation of names as character strings.

The information type domain object information is used to define the structure of a design object. It is available within both basic design object information and derivable design object information. Several sorts and objects of sorts have been added to construct the following relations:

**subsorts**
component_name,
information_link_name,
knowledge_base_name,
information_type_name,
sort_name,
object_name,
function_name,
relation_name: specific_name;

**relations**
is_toplevel,
is_component: component_name;

```
has_subcomponent:                  component_name *
                                   component_name;
has_interface_information_type:    component_name *
                                   input_output *
                                   information_type_name;
is_information_link:               information_link_name;
has_information_link:              component_name *
                                   information_link_name;
has_source_component,
has_destination_component:         information_link_name *
                                   component_name;
has_source_information_type,
has_destination_information_type:  information_link_name *
                                   information_type_name;
corresponds_with:                  specific_name *
                                   user_given_name;
has_knowledge_base:                component_name *
                                   knowledge_base_name;
refers_to_information_type:        information_type_name *
                                   information_type_name;
refers_to_information_type:        knowledge_base_name *
                                   information_type_name;
refers_to_metadescription_of:      information_type_name *
                                   sort_name *
                                   information_type_name;
```

A compositional system can be described using the relations described above.

The information type derivable domain object information is extended with relations with which properties of compositional systems can be described. The relation has_property is defined as follows:

```
functions
    is_capable_of_bidirectional_communication_with:   component_name ->
                                                      property_of_an_agent;
subsorts
    property_of_a_diagnostic_system,
    property_of_an_agent,
    property_of_the_material_world,
    property_of_the_multi_agent_system:   property;
    diagnostic_reasoning_properties,
    strategy_determination_properties:    property_of_a_diagnostic_system;
    communication_properties,
    cooperation_properties,
    interaction_properties,
    own_process_control_properties:       property_of_an_agent;
relations
    has_property:                         specific_name *
                                          property;
```

The specification above defines the properties of a diagnostic reasoning system; the properties of agents distinguished in Chapter 5 are represented as shown above.

The information types basic design requirement information and derivable design requirement information defined in the generic design model are also intended to be instantiated. The definition of design requirement information already includes meta-descriptions of basic and derivable design object information which is thus available in basic design requirement information and derivable design requirement information. In this specialisation the information type basic design requirement information is extended with relations with which refinement relationship among qualified requirements can be described:

```
relations
    has_been_refined_by:   qualified_requirement_name *
                           qualified_requirement_name;
```

The information type derivable design requirement information is extended with relations with which properties of design requirements can be described:

```
sorts
    conflict_type;
objects
    …,
    multi_agent_system_principles:      conflict_type;
relations
    have_conflict_on:                   list_of_qualified_requirement_name *
                                        conflict_type;
    has_refinement,
    is_a_refinement:                    qualified_requirement_name;
    qr_structural_influence:            qualified_requirement_name *
                                        structural_influence;
```

The relation have conflict on denotes the type of conflict detected between a number of qualified requirements (e.g., violating multi-agent system principles: a component cannot be an agent and an external world). The relation has refinement and is a refinement are statements on individual qualified requirements which express whether a qualified requirement has one or more refinements, and whether a qualified requirement is a refinement of another qualified requirement. The relation qr structural influence specifies the structural influence of a particular qualified requirement. The structural influence is restricted to knowledge composition, i.e. whether a qualified requirement is only concerned with knowledge composition, and not process composition (of the system to be re-designed).

## 7.2 Refinement of design process co-ordination

The co-ordination of the overall design process (DPC) is briefly described in the generic model of design in Section 6.1.1. Section 7.2.1 addresses a specialisation of the process composition of DPC and Section 7.2.2 focuses on a specialisation of the knowledge composition of DPC.

### 7.2.1 Process refinement of DPC

Specialisation of the process composition of DPC includes identification of more specific processes at lower levels of (process) abstraction, and the relation between processes in terms of lower level processes.

**Identification of processes at different levels of abstraction.** The process of DPC determines design strategies, on the basis of design process objectives and previous results of the design process. To this purpose a number of sub-processes may be required such as shown in Figure 7.3. The process DPC determines overall design strategies and resource restrictions for RQSM and DODM.

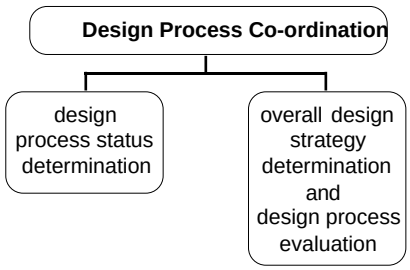


**Figure 7.3** Processes at different abstraction levels within DPC.

The process process status determination is responsible for gathering information on achievements of the design process, with respect to previous design strategies. The process design process status determination determines, on the basis of the information gathered and

design process objectives, the current status of the design process. The process overall design strategy determination and design process evaluation is responsible for the formulation of design strategies to be provided and evaluates the design process objectives, on the basis of results from design process status determination.

Each of the sub-processes of DPC depicted in Figure 7.3 can be characterised in terms of their input and output information types, as shown in Table 7.1.

| *DPC* | *input information type* | *output information type* |
|---|---|---|
| design process status determination | • manipulation process evaluation<br>• previous overall design strategy<br>• design process objectives | • design process status |
| overall design strategy determination and design process evaluation | • design process status<br>• design process objectives | • overall design strategy<br>• design process evaluation |

**Table 7.1** Input and output information types of processes within design process co-ordination.

Table 7.1 depicts the following input and output information in the interface of the processes in DPC:

- The component design process status determination requires evaluations of the manipulation processes (manipulation process evaluation), the overall design strategy on which the reports are based (previous overall design strategy), and the given objectives on the design process (design process objectives). This component produces an analysis of the current state of the design process (design process status).
- The component overall design strategy determination and design process evaluation needs an analysis of the current state of the design process (design process status) and the given objectives on the design process (design process objectives). This component produces an design strategy for both manipulation processes (overall design strategy) and an evaluation of the design process (design process evaluation).

The next section describes the composition relation between these processes.

**Composition of processes.** Both static and dynamic perspectives on process composition are of importance. In this section information exchange is first addressed (static perspective) and then task control (dynamic perspective).

The static perspective on the composition relation defines the *information links* in the process design process co-ordination as shown in Figure 7.4. The following information links are shown in Figure 7.4:

- The mediating link manipulation process evaluation transfers information on RQSM and DODM evaluations of previous design strategies (manipulation process evaluation) from the input interface of DPC to the input interface of design process status determination.
- The private link previous overall design strategy transfers the overall design strategy (overall design strategy) from the output interface of overall design strategy determination and design process evaluation to previous overall design strategy in the input interface of the design process status determination on the basis of an explicit mapping between these information types.
- The mediating link design process objectives to status determination transfers the design process objectives (design process objectives) from the output interface of DPC to the input interface of design process status determination.
- The mediating link design process evaluation transfers an evaluation of the design process objectives (design process evaluation) from the output interface of overall design strategy determination and design process evaluation to the output interface of DPC.
- The private link design process status transfers design process status from the output interface of design process status determination to the input interface of overall design strategy determination and design process evaluation.

- The mediating link design process objectives transfers the design process objectives (design process objectives) from the input interface of DPC to the input interface of overall design strategy determination and design process evaluation.
- The mediating link overall design strategy transfers information on the current overall design strategy (overall design strategy) from the output interface of overall design strategy determination and design process evaluation to the output interface of DPC.

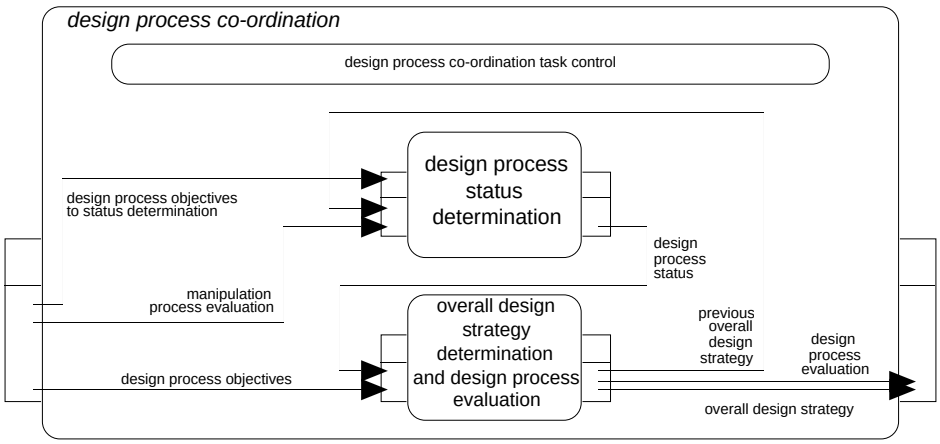


**Figure 7.4** Information links within the process of design process co-ordination.

The dynamic perspective on the composition relation defines control over the sub-components of the component design process co-ordination. The task control within DPC defines two phases: establishing the status of the design process, and issuing overall design strategies or termination of the design process.

Upon activation of the component DPC, the component design process status determination is activated and the information links manipulation process evaluation, design process objectives to status determination, and previous overall design strategy are made up-to-date. Upon termination of the component design process status determination, the component overall design strategy determination and design process evaluation is activated. The information links design process status and design process objectives are made up-to-date. Upon termination of the component overall design strategy determination and design process evaluation, the component DPC is terminated and the information links overall design strategy, and design process evaluation are made up-to-date.

### 7.2.2 Knowledge refinement related to DPC

Specialisation of the knowledge of DPC identifies specific knowledge structures at different levels of (knowledge ) abstraction, and describes how a knowledge structure is defined in terms of lower level knowledge structures.

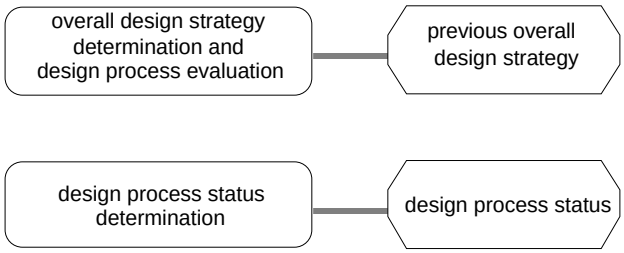


**Figure 7.5** Two information types employed in interfaces of sub-processes of DPC.

**Identification of knowledge structures at different levels of abstraction.** The *information types* design process objectives, manipulation process evaluation, design process

evaluation, and overall design strategy are described in Chapter 6 and Appendix A.1. The information types previous overall design strategy, and design process status are shown in Figure 7.5.

The relation defined in the information type previous overall design strategy is:

```
relations
    is_old_design_strategy:        design_strategy_name *
                                   manipulation_process_property_expression;
```

The relation is old design strategy denotes a previous instance of the relation is design strategy (see Section A.1.5): each strategy has a name, and expressions on properties of the manipulation processes.

Relations defined in the information type design process status are:

```
relations
    RQSM_progress_status,
    DODM_progress_status:          progress_status;
    information_available_for_DODM,
    information_available_for_RQSM;
```

The relations RQSM progress status and DODM progress status have the sort progress status, which contains objects such as complete success, complete failure, no action required, partial success, and partial failure, as their arguments. The sort progress status is part of the generic model of design, and has not been specialised. The relations information available for DODM and information available for RQSM denote whether information is available which can be used by the processes DODM and RQSM.

A *knowledge base* defines the knowledge used in one or more processes. Relations between information types and knowledge bases specify the ontology used within a knowledge base.

The knowledge bases design process status determination knowledge, and overall design strategy determination and design process evaluation knowledge are distinguished, as shown in Figure 7.6.

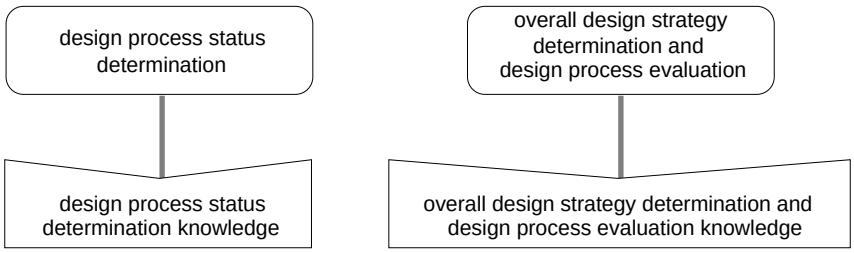


**Figure 7.6** Knowledge bases corresponding to sub-processes of DPC.

Example overall design strategy determination and design process evaluation knowledge is given below:

```
if   is_qualified_process_objective(   QN: qualified_process_objective_name,
                                       obligatory,
                                       NO: process_objective_name )
     and   is_process_objective(       NO: process_objective_name,
                                       start_process )
     then  is_design_strategy(         initiate_new_re_design_process );

if   is_qualified_process_objective(   QN: qualified_process_objective_name,
                                       obligatory,
                                       NO: process_objective_name )
     and   is_process_objective(       NO: objective_name,
                                       is_RQS_to_be_used( This: RQS_name ) )
     then  is_design_strategy(         focus_manipulation_on( This: RQS_name ) );
```

**Composition of knowledge structures.** The composition of knowledge related to the component DPC is described in this section. The information types described in this section are

not shown to refer to other information types. One of the knowledge bases described in this section, overall design strategy determination and design process evaluation knowledge, is related to a number of information types, as shown in Figure 7.7.

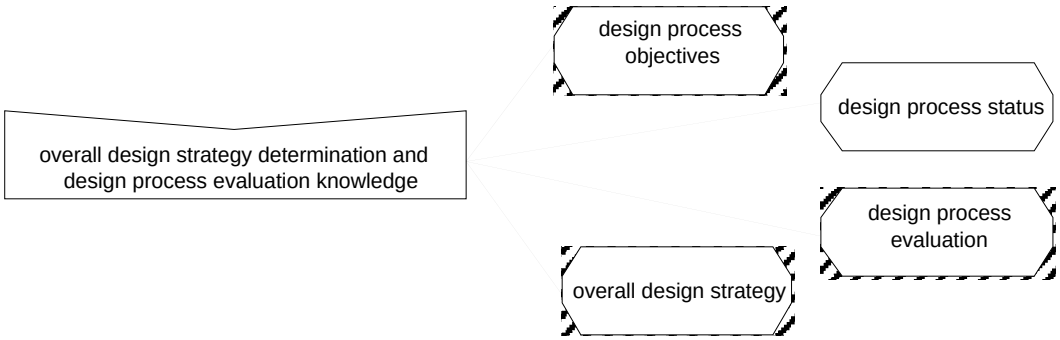


**Figure 7.7** Relation between the knowledge base overall design strategy determination and design process evaluation knowledge and a number of information types.

## 7.3 Discussion

In this chapter a refinement of the generic model for design (Chapter 6; Brazier, Langen, Ruttkay and Treur, 1994) has been described. The process composition and knowledge composition of the generic model of design has been partially refined for the domain of 're-design of compositional systems': the information types related to the interface of the design process are instantiated and the process design process co-ordination is refined. The representation of the structure of a compositional system and properties of a compositional system within a design system has been described and illustrated with examples of knowledge.

The refinement steps described in this chapter are depicted in Figure 7.8. The first refinement step is described in Section 7.1: information types related to the interface of the design process are instantiated. The second refinement step is described in Section 7.2: the process design process co-ordination is refined.

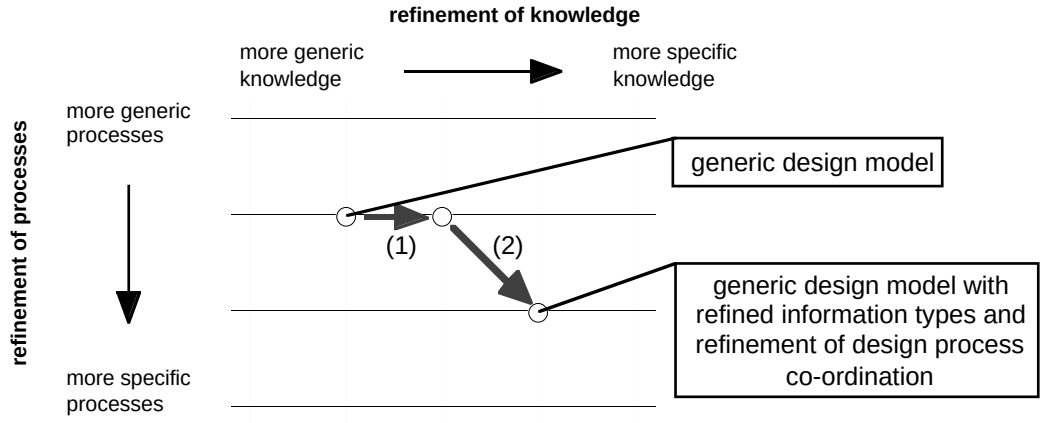


**Figure 7.8** Refinement steps described in this chapter: the numbered arrows correspond to sections 7.1 to 7.2.

Table 7.2 describes which part of the current refinement of the generic design model provides the realisation for which desideratum.

| *desideratum* | *is realised by* |
|---|---|
| **dr1** | instantiation of information types domain object information, basic domain object information, and derivable domain object information (see Section 7.1). |
| **dr2** | instantiation of information types design requirement information, basic design requirement information, and derivable design requirement information (see Section 7.1). |
| **dr5** | refinement of component design process co-ordination (see Section 7.2). |

**Table 7.2** Desiderata realised by specific parts of the refinement of the design model.

The process composition and knowledge composition described in this chapter are generic in the sense that more elaborate, knowledge-intensive specialisations can be added. For example, in the refinement of design process co-ordination, a more elaborate specialisation could employ, for example, a history (e.g. for searching for previously successful design strategies).

In the next two chapters additional refinements of the generic design model are described. First refinements of the process RQS manipulation are presented, then refinements of the process DOD manipulation.

# 8 Requirement Qualification Set Manipulation in the Re-design Model for Compositional Systems

In this chapter additional refinements of the generic design model are described on the basis of the results of Chapter 7. The realisation of the following desideratum is addressed in this chapter:

- dr4, model of the manipulation of requirements on compositional systems.

The process RQS manipulation (RQSM) determines which modifications to a set of design requirements are most appropriate on the basis of information on overall design strategies, and information from DODM. The process of RQSM is composed of four sub-processes (see Figure 6.13): RQSM history maintenance, RQS modification, current RQS Maintenance, and deductive RQS refinement. The first two sub-processes are composed, the last two are not. Section 8.1 describes a refinement of current RQS maintenance, Section 8.2 describes a refinement of deductive RQS refinement, Section 8.3 describes a refinement of RQS modification, and Section 8.4 describes a refinement of RQSM history maintenance. In each of these sections both process composition and knowledge composition are addressed. In Section 8.5 the refinement of the process RQS manipulation is briefly discussed.

## 8.1 Refinement of current RQS maintenance

The process current RQS maintenance is not composed. The information types in the interface of current RQS maintenance, discussed in Section 6.1.2, are specialised as described in Section 7.1. Knowledge employed in this process can also be instantiated for the domain of re-design of compositional systems. The process current RQS maintenance can employ knowledge to, e.g., distinguish viewpoints: a focus can be made, highlighting all design requirements corresponding to a particular viewpoint. An instance of knowledge used to focus on design requirements is given below.

**Example from knowledge base current RQS maintenance knowledge**

The knowledge element below has the current focus as its first condition. Design requirements corresponding to this focus need to be identified. The second condition identifies a qualified requirement and its related requirement. The third condition ascertains that the requirement is related to a property which is in focus. The conclusion states that this qualified requirements is in the current focus.

```
if   current_focus(                  multi_agent_system_properties )
     and  is_qualified_requirement(  QRN: qualified_requirement_name,
                                     Q: qualification,
                                     RN: requirement_name )
     and  is_requirement(            RN: requirement_name,
                                     has_property(   C: component_name,
                                                     P: multi_agent_system_property ) )
then      in_current_focus(          QRN: qualified_requirement_name );
```

## 8.2 Refinement of deductive RQS refinement

The process deductive RQS refinement is not composed. The information types in the interface of deductive RQS refinement, discussed in Section 6.1.2, are specialised as described in Section

7.1. Knowledge employed in this process can also be instantiated for the domain of re-design of compositional systems. The process deductive RQS refinement can employ knowledge to, e.g., derive properties of design requirements: conflicting design requirements can be detected. An example of knowledge to detect a conflict between design requirements is given below.

**Example from knowledge base deductive RQS refinement knowledge**

The knowledge element below has two qualified requirements and two unqualified requirements as its conditions. The first qualified requirement is a requirement which refers to the expression that a component C has the property is capable of bi-directional communication with a component D. The second qualified requirement is a requirement which refers to the expression that a component D has a material world property. This leads to the conclusion that these qualified requirements are in conflict with each other: one of the multi-agent system principles is violated: a material world is *not* an agent, therefore a material world cannot have agent related properties (see Section 5.3 for a description of properties of a multi-agent system).

```
if   is_qualified_requirement(   QRN: qualified_requirement_name,
                                 Q1: qualification,
                                 N: requirement_name )
     and   is_requirement(       N: requirement_name,
                                 has_property(       C: component_name,
                                                     is_capable_of_bidirectional_communication_with(
                                                     D: component_name ) ) )
     and   is_qualified_requirement( QRM: qualified_requirement_name,
                                     Q2: qualification,
                                     M: requirement_name )
     and   is_requirement(       M: requirement_name,
                                 has_property(       D: component_name,
                                                     W: property_of_the_material_world ) )
then      have_conflict_on(  [ QRN: qualified_requirement_name,
                               QRM: qualified_requirement_name ],
                           multi_agent_system_principles );
```

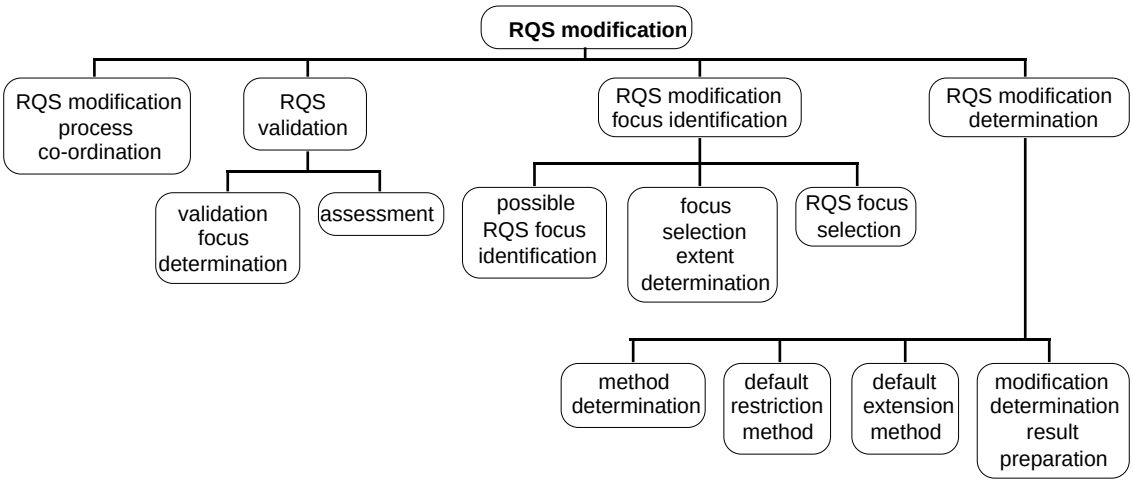


**Figure 8.1** Partial process refinement for RQS modification.

## 8.3 Refinement of RQS Modification

For the process RQS modification first the process composition is described, then the knowledge composition is described.

### 8.3.1 Process composition within RQS modification: identification of processes and abstraction levels

The process RQS modification determines modifications to a requirement qualification set (RQS). To this purpose a number of sub-processes are distinguished as shown in Figure 8.1. The process RQS modification process co-ordination is responsible for the co-ordination of the entire process within RQSM: this process determines whether, when, and by which means, a particular RQS is to be modified.

The global phases within RQS modification resemble a process control model (e.g., controlling a chemical process). In a process control task a cycle occurs over the sub-tasks: analysis, planning, execution. Similarly, within RQS modification analysis is performed by RQS validation, planning is performed by RQS modification focus identification and RQS modification determination, and execution is performed by effectuating modifications to a RQS, resulting in a new RQS in current RQS maintenance.

The process RQS modification process co-ordination co-ordinates the modification process on the basis of the overall design strategy and information available in the RQSM history. A modification strategy is issued, which influences the other three sub-processes of RQS modification. On the basis of results obtained from these three sub-processes, additional modification strategies can be issued, histories can be inspected, information can be stored in the history, or the RQS modification process can be terminated. This process is described in more detail in Appendix B.1.1.

The process RQS validation validates the current RQS. Its sub-process validation focus determination determines which properties of (qualified) requirements need to be validated (e.g. apparent conflicts, aggregation level per (qualified) requirement, etc.). The process deductive RQS refinement (see Figure 6.13) is given these results as goals to pursue. The sub-process assessment assesses the achievement of the validation focus on the basis of the results of the deductive refinement.

The process RQS modification focus identification determines which (qualified) requirements need to be modified, on the basis of a given internal strategy. To identify a focus, three processes are distinguished: possible RQS focus identification identifies one or more candidate foci. On the basis of the current RQS modification strategy the process focus selection extent determination decides whether one or all of the candidate foci are to be selected as the current focus or foci. The process RQS focus selection then selects one (or more) foci from the candidate foci.

The process RQS modification determination determines the actual modifications to the current RQS on the basis of given strategies from RQS modification process co-ordination, and information on the modification focus from RQS modification focus identification. This process entails four sub-processes. The process method determination chooses the best method corresponding to the given strategy, given the current information. The process default restriction method determines possible modifications which remove certain design requirements, on the basis of the current modification focus. The process default extension method determines possible modifications which add an appropriate refined design requirement on the basis of the current modification focus. The process modification determination result preparation analyses the proposed modifications, e.g., to ascertain non-duplicate design requirements, before formulating the final modifications to the current RQS. The process default extension method is described in more detail in Appendix B.1.2.

The interface information types for sub-processes of RQS modification are listed in Table 8.1 and described below.

- The process RQS modification process co-ordination needs overall design strategy (overall design strategy), results of searching the RQS modification state history (RQS modification state history search results), results of searching the RQS assessment history (RQS assessment history search results), results of searching the DOD assessment history (DOD assessment history search results), results of searching the RQS history (RQS history search results), results on the success of a replacement request (current RQS replacement request), modification foci (RQS modification focus), evaluations of individual design

requirements (current RQS basis evaluation), and modifications (RQS modification information). This process generates information on the progress of the modification process (RQS modification progress), actions for the manipulation process (current manipulation action), queries on RQS modification state history ( RQS modification state history queries), an evaluation of resulting requirement qualification sets (RQS assessment), queries on RQS assessment history (RQS assessment history queries), queries on DOD assessment history (DOD assessment history queries), queries on RQS history (RQS history queries), modification strategy (RQS modification strategy), and request for replacement of the current RQS (current RQS replacement request).

- The process RQS validation requires the contents of the current RQS (current RQS contents), and produces goals to achieve for deductive refinement of a RQS (RQS refinement goals) and the assessment of individual design requirements (current RQS basis evaluation).
- The process RQS modification focus identification requires a modification strategy (RQS modification strategy), assessment of individual design requirements (current RQS basis evaluation), and the contents of the current RQS (current RQS contents). The process RQS modification focus identification generates modification foci (RQS modification focus).
- The process RQS modification determination requires the modification strategy (RQS modification strategy), assessment of individual design requirements (current RQS basis evaluation), modification foci (RQS modification focus), and the contents of the current RQS (current RQS contents). The results of this process are an indication of the status of the modification process (modification status), and modifications to be performed on the current RQS (RQS modifications).

| *process* | *input information type* | *output information type* |
|---|---|---|
| RQS modification process co-ordination | • overall design strategy<br>• RQS modification state history search results<br>• RQS assessment history search results<br>• DOD assessment history search results<br>• RQS history search results<br>• current RQS replacement results<br>• RQS modification focus<br>• RQS assessment<br>• RQS modification information | • RQS modification progress<br>• current manipulation action<br>• RQS modification state history queries<br>• RQS assessment<br>• RQS assessment history queries<br>• DOD assessment history queries<br>• RQS history queries<br>• RQS modification strategy<br>• current RQS replacement request |
| RQS Validation | • current RQS contents | • RQS refinement goals<br>• current RQS basis evaluation |
| RQS Modification Focus Identification | • RQS modification strategy<br>• current RQS basis evaluation<br>• current RQS contents | • RQS modification focus |
| RQS Modification Determination | • RQS modification strategy<br>• current RQS basis evaluation<br>• RQS modification focus<br>• current RQS contents | • modification status<br>• RQS modifications |

**Table 8.1** Interface information types for sub-processes of RQS modification.

| *process* | *input information type* | *output information type* |
|---|---|---|
| validation focus determination | • current RQS contents | • RQS refinement goals |
| assessment | • current RQS contents<br>• RQS refinement goals | • current RQS basis evaluation |

**Table 8.2** Interface information types for sub-processes of RQS validation.

The interface information types of the sub-processes of RQS validation are listed in Table 8.2 and described below.

- The process validation focus determination requires information on the contents of the current RQS (current RQS contents) and produces information on goals to achieve for deductive refinement of a RQS (RQS refinement goals).
- The process assessment requires information on the contents of the current RQS (current RQS contents) and goals to achieve for deductive refinement of a RQS (RQS refinement goals), and produces information on the assessment of individual design requirements (current RQS basis evaluation).

| *process* | *input information type* | *output information type* |
|---|---|---|
| possible RQS focus identification | • RQS modification strategy<br>• current RQS contents<br>• current RQS basis evaluation | • possible RQS modification focus |
| focus selection extent determination | • RQS modification strategy | • focus selection extent |
| RQS focus selection | • RQS modification strategy<br>• possible RQS modification focus | • selected RQS modification focus |

**Table 8.3** Interface information types for sub-processes of RQS focus identification.

The interface information types of the sub-processes of RQS modification focus identification are listed in Table 8.3 and described below.

- The process possible RQS focus identification requires a modification strategy (RQS modification strategy), assessment of individual design requirements (current RQS basis evaluation), and the contents of the current RQS (current RQS contents). As its results, it generates possible RQS modification foci (possible RQS modification focus).
- The process focus selection extent determination needs a modification strategy (RQS modification strategy), and produces an indication of the extent of selections of modification foci (focus selection extent).
- The process RQS focus selection needs a modification strategy (RQS modification strategy), and possible RQS modification foci (possible RQS modification focus). This process produces selected RQS modification foci (selected RQS modification focus).

| *process* | *input information type* | *output information type* |
|---|---|---|
| method determination | • RQS modification strategy | • modification method<br>• modification status |
| default restriction method | • modification method<br>• RQS modification focus<br>• current RQS contents<br>• rejected RQS modifications | • restriction results |
| default extension method | • modification method<br>• RQS modification strategy<br>• RQS modification focus<br>• current RQS contents<br>• rejected RQS modifications | • extension results |
| modification determination result preparation | • current RQS contents<br>• restriction results<br>• extension results | • selected RQS modifications |

**Table 8.4** Interface information types for sub-processes of RQS modification determination.

The interface information types of the sub-processes of RQS modification determination are listed in Table 8.4 and described below.

- The process method determination requires a modification strategy (RQS modification strategy). This process generates a method for modification (modification method), and a status of this modification process (modification status).
- The process default restriction method needs a method for modification (modification method), a focus for modification (RQS modification focus), the contents of the current RQS

(current RQS contents), and rejected modifications on the current RQS (rejected RQS modifications). Its results are restrictions on the current RQS (restriction results).
- The process default extension method requires a method for modification (modification method), a modification strategy (RQS modification strategy), a focus for modification and modification limitations (RQS modification focus), the contents of the current RQS (current RQS contents), and rejected RQS modifications (rejected RQS modifications). This process produces extensions on the current RQS (extension results).
- The process modification determination result preparation needs the contents of the current RQS (current RQS contents), restrictions on the current RQS (restriction results), and extensions on the current RQS (extension results). Its results are selected modifications on the current RQS (selected RQS modifications).

### 8.3.2 Process composition relation within RQS modification

The *information links*, i.e., the static perspective on the process composition, is described for the processes identified above. The information links in the component RQS modification are shown in Figure 8.2.

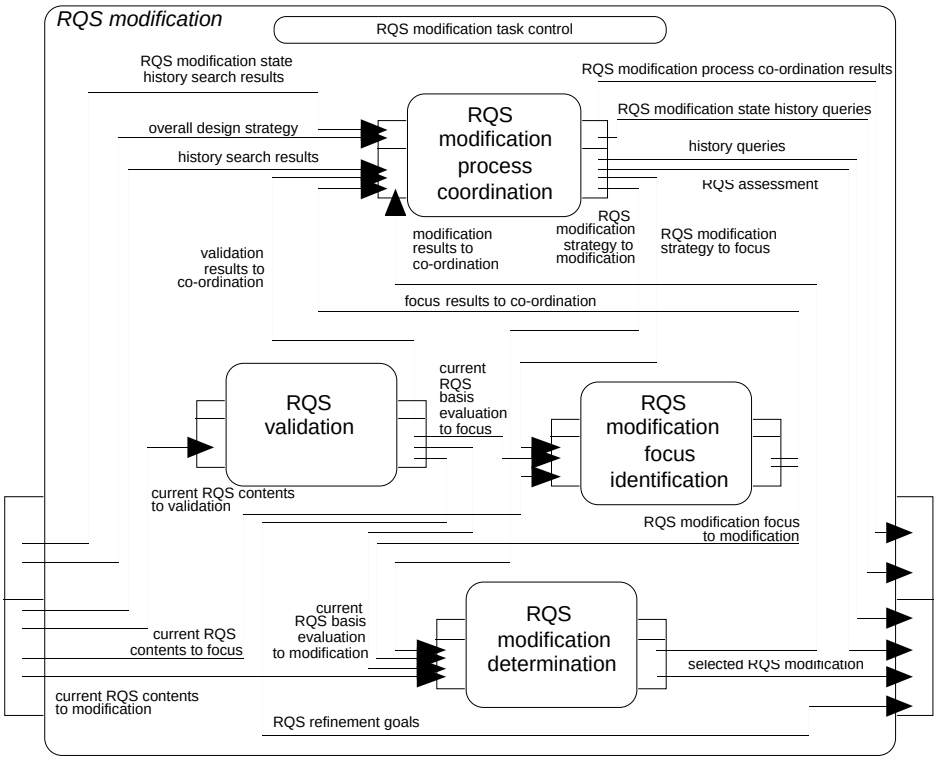


**Figure 8.2** Information links within the process of RQS modification.

Within this component twelve mediating links and eight private links are defined:

- The mediating link RQS modification state history search results transfers the information expressed in RQS modification state history search results from the input interface of RQS modification to the input interface of RQS modification process co-ordination.
- The mediating link overall design strategy transfers the overall design strategy (overall design strategy) from the input interface of RQS modification to the input interface of RQS modification process co-ordination.
- The mediating link history search results transfers the information expressed in RQS assessment history search results, DOD assessment history search results, RQS history search results and current RQS replacement results the input interface of RQS modification to the input interface of RQS modification process co-ordination.

- The mediating link current RQS contents to focus transfers the contents of the current RQS (current RQS contents) from the input interface of RQS modification to the input interface of RQS modification focus identification.
- The mediating link current RQS contents to validation transfers the contents of the current RQS (current RQS contents) from the input interface of RQS modification to the input interface of RQS validation.
- The mediating link current RQS contents to modification transfers the contents of the current RQS (current RQS contents) from the input interface of RQS modification to the input interface of RQS modification determination.
- The private link current RQS basis evaluation to focus transfers assessments of individual design requirements (current RQS basis evaluation) from the output interface of RQS validation to the input interface of RQS modification focus identification.
- The private link current RQS basis evaluation to modification transfers assessments of individual design requirements (current RQS basis evaluation) from the output interface of RQS validation to the input interface of RQS modification determination.
- The private link RQS modification focus transfers foci for modification (RQS modification focus) from the output interface of RQS modification focus identification to the input interface of RQS modification determination.
- The private link validation results to co-ordination transfers individual design requirement assessments (current RQS basis evaluation) from the output interface of RQS validation to the input interface of RQS modification process co-ordination.
- The private link focus results to co-ordination transfers modification foci (RQS modification focus) from the output interface of RQS modification focus identification to the input interface of RQS modification process co-ordination.
- The private link modification results to co-ordination transfers modifications to the current RQS (RQS modification information) and modification method status (modification status) from the output interface of RQS modification determination to the input interface of RQS modification process co-ordination.
- The private link RQS modification strategy to focus transfers modification strategy (RQS modification strategy) from the output interface of RQS modification process co-ordination to the input interface of RQS modification focus identification.
- The private link RQS modification strategy to modification transfers modification strategy (RQS modification strategy) from the output interface of RQS modification process co-ordination to the input interface of RQS modification determination.
- The mediating link selected RQS modification transfers modifications to be performed (selected RQS modification) from the output interface of RQS modification determination to the output interface of RQS modification.
- The mediating link RQS refinement goals transfers goals for deductive refinement of the contents of a RQS (RQS refinement goals) from the output interface of RQS validation to the output interface of RQS modification.
- The mediating link RQS modification process co-ordination results transfers information on the progress of the modification process (RQS modification progress), and the current manipulation action (current manipulation action) from the output interface of RQS modification process co-ordination to the output interface of RQS modification.
- The mediating link RQS modification state history queries transfers queries on the RQS modification state historical ( RQS modification state history queries) from the output interface of RQS modification process co-ordination to the output interface of RQS modification.
- The mediating link history queries transfers queries on the RQS assessment history (RQS assessment history queries), queries on the DOD assessment history (DOD assessment history queries), queries on RQS history (RQS history queries), and requests for replacement of the current RQS (current RQS replacement request) from the output interface of RQS modification process co-ordination to the output interface of RQS modification.

- The mediating link RQS assessment transfers results of the modification process (RQS assessment) from the output interface of RQS modification process co-ordination to the output interface of RQS modification.

The task control within the component RQS modification distinguishes a number of phases, which correspond to manipulation actions determined by RQS modification process co-ordination. Upon activation of RQS modification, RQS modification process co-ordination is activated. A distinction can be made between continuation of the previous manipulation process, or initiation of a new manipulation process. Queries on history, requests for replacement of the current RQS, and information on the modification process can be produced by RQS modification process co-ordination. RQS validation can be activated to determine refinement goals, and to evaluate design requirements on the basis of refinement results. The modification strategy, also produced by RQS modification process co-ordination can be made available to both RQS modification focus identification and RQS modification determination, which are activated in that order. Selected modifications to the current RQS can be made available as output of RQS modification. A manipulation action determined by RQS modification process co-ordination indicates that RQS modification can terminate itself.

### 8.3.3 Process composition relation within RQS validation

The information links in the component RQS validation are shown in Figure 8.3.

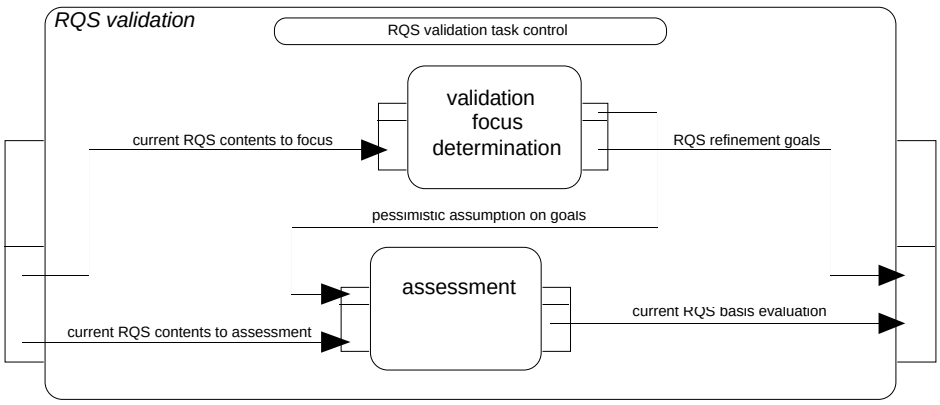


**Figure 8.3** Information links within the process of RQS validation.

Within this component four mediating links and one private link are defined:

- The mediating link current RQS contents to focus transfers the contents of the current RQS (current RQS contents) from the input interface of RQS validation to the input interface of validation focus determination.
- The mediating link current RQS contents to assessment transfers the contents of the current RQS (current RQS contents) from the input interface of RQS validation to the input interface of assessment.
- The private link pessimistic assumption on goals transfers the occurrence of goals for deductive refinement of a RQS (epistemic of RQS refinement goals) from the output interface of validation focus determination to assumptions on not having achieved these goals (assumption on current RQS contents) in the input interface of assessment on the basis of an explicit mapping between these information types.
- The mediating link RQS refinement goals transfers goals for deductive refinement of the contents of a RQS (RQS refinement goals) from the output interface of validation focus determination to the output interface of RQS validation.
- The mediating link current RQS basis evaluation transfers assessments of individual design requirements (current RQS basis evaluation) from the output interface of assessment to the output interface of RQS validation.

The task control within the component RQS validation is as follows. Upon activation of RQS validation a number of possible situations are possible, which correspond to task control foci for this component. The task control foci are: prepare for validation, and assess validation results. For each of these task control foci specific links need to be made up-to-date:

- On activation with the task control focus prepare for validation the information link current RQS contents to focus is made up-to-date and validation focus determination is activated. Upon termination of validation focus determination, the information link RQS refinement goals is made up-to-date and RQS validation terminates itself.
- On activation with the task control focus assess validation results the information links current RQS contents to assessment and pessimistic assumption on goals are made up-to-date and assessment is activated. Upon termination of assessment, the information link current RQS basis evaluation is made up-to-date and RQS validation terminates itself.

### 8.3.4 Process composition relation within RQS modification focus identification

The information links in the component RQS modification focus identification are shown in Figure 8.4.

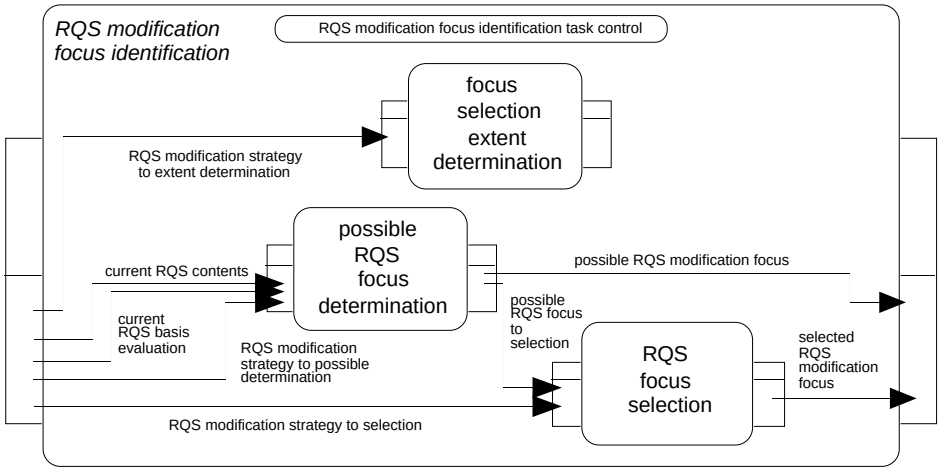


**Figure 8.4** Information links within the process of RQS modification focus identification.

Within this component seven mediating links and one private link are defined:

- The mediating link RQS modification strategy to extent determination transfers the modification strategy (RQS modification strategy) from the input interface of RQS modification focus identification to the input interface of focus selection extent determination.
- The mediating link current RQS contents transfers the contents of the current RQS (current RQS contents) from the input interface of RQS modification focus identification to the input interface of possible RQS focus determination.
- The mediating link current RQS basis evaluation transfers assessments of individual design requirements (current RQS basis evaluation) from the input interface of RQS modification focus identification to the input interface of possible RQS focus determination.
- The mediating link RQS modification strategy to possible determination transfers the modification strategy (RQS modification strategy) from the input interface of RQS modification focus identification to the input interface of possible RQS focus determination.
- The mediating link RQS modification strategy to selection transfers the modification strategy (RQS modification strategy) from the input interface of RQS modification focus identification to the input interface of RQS focus selection.
- The private link possible RQS focus to selection transfers possible foci for modification (possible modification focus) from the output interface of possible RQS focus determination to the input interface of RQS focus selection.

- The mediating link possible RQS modification focus transfers possible foci for modification (possible modification focus) from the output interface of possible RQS focus determination to the output interface of RQS modification focus identification.
- The mediating link selected RQS modification focus transfers selected foci for modification (selected modification focus) from the output interface of RQS focus selection to the output interface of RQS modification focus identification.

The task control within the component RQS modification focus identification is as follows. Upon activation of RQS modification focus identification, the information links current RQS contents, current RQS basis evaluation, and RQS modification strategy to possible determination are made up-to-date and possible RQS focus determination is activated. Upon termination of possible RQS focus determination the information link RQS modification strategy to extent determination is made up-to-date and focus selection extent determination is activated. Upon termination of focus selection extent determination, one of several evaluation criteria becomes successful, which indicates with which extent RQS focus selection needs to be activated. The information links possible RQS focus to selection, and RQS modification strategy to selection are made up-to-date and RQS focus selection is activated with a particular extent (any new, or all possible). Upon termination of RQS focus selection the information links possible RQS modification focus and selected RQS modification focus are made up-to-date and RQS modification focus identification terminates itself.

### 8.3.5 Process composition relation within RQS modification determination

The information links in the component RQS modification determination are shown in Figure 8.5.

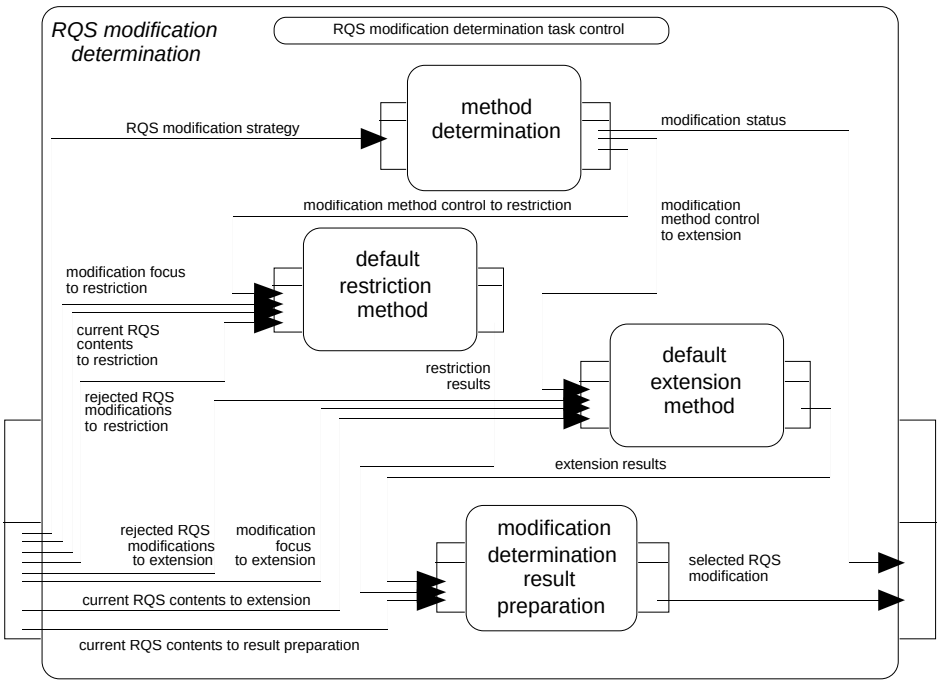


**Figure 8.5** Information links within the process of RQS modification determination.

Within this component ten mediating links and four private links are defined:

- The mediating link RQS modification strategy transfers the modification strategy (RQS modification strategy) from the input interface of RQS modification determination to the input interface of method determination.
- The mediating link current RQS contents to restriction transfers the contents of the current RQS (current RQS contents) from the input interface of RQS modification determination to the input interface of default restriction method.

- The mediating link modification focus to restriction transfers the focus for modification (RQS modification focus) from the input interface of RQS modification determination to the input interface of default restriction method.
- The mediating link rejected RQS modifications to restriction transfers rejected foci for modification (rejected RQS modification focus) from the input interface of RQS modification determination to the input interface of default restriction method.
- The mediating link modification focus to extension transfers the focus for modification (RQS modification focus) from the input interface of RQS modification determination to the input interface of default extension method.
- The mediating link rejected modifications to extension transfers rejected foci for modification (rejected RQS modification focus) from the input interface of RQS modification determination to the input interface of default extension method.
- The mediating link current RQS contents to extension transfers the contents of the current RQS (current RQS contents) from the input interface of RQS modification determination to the input interface of default extension method.
- The mediating link current RQS contents to result preparation transfers the contents of the current RQS (current RQS contents) from the input interface of RQS modification determination to the input interface of modification determination result preparation.
- The private link modification method control to restriction transfers control information for modification methods (modification method) from the output interface of method determination to the input interface of default restriction method.
- The private link modification method control to extension transfers control information for modification methods (modification method) from the output interface of method determination to the input interface of default extension method.
- The private link restriction results transfers results from the restriction method (restriction results) from the output interface of default restriction method to the input interface of modification determination result preparation.
- The private link extension results transfers results from the extension method (extension results) from the output interface of default extension method to the input interface of modification determination result preparation.
- The mediating link modification status transfers status information on modification methods (modification status) from the output interface of method determination to the output interface of RQS modification determination.
- The mediating link selected RQS modification transfers selected modifications of a RQS (selected RQS modification) from the output interface of modification determination result preparation to the output interface of RQS modification determination.

The task control within the component RQS modification determination is as follows. Upon activation of RQS modification determination the information link RQS modification strategy is made up-to-date and method determination is activated. Upon termination of method determination one of two evaluation criteria is successful: restriction sub-task is to be performed next, or extension sub-task is to be performed next. In each case, a different sub-task is activated.

- On activation with task control focus restriction sub-task is to be performed next the information links modification method control to restriction, modification focus to restriction, current RQS contents to restriction, and rejected RQS modifications to restriction are made up-to-date and default restriction method is activated. Upon termination of default restriction method, the information links restriction results and current RQS contents to result preparation are made up-to-date and modification determination result preparation is activated
- On activation with task control focus extension sub-task is to be performed next, the information links modification method control to extension, modification focus to extension, rejected RQS modifications to extension, and current RQS contents to extension are made up-to-date and default extension method is activated. Upon termination of default extension

method the information links extension results and current RQS contents to result preparation are made up-to-date and modification method result preparation is activated.

Upon termination of modification determination result preparation the information links modification status and selected RQS modification are made up-to-date and RQS modification determination terminates itself.

### 8.3.6 Knowledge composition for RQS modification

A number of information types and knowledge bases related to sub-processes of RQS modification are described below. The information type current RQS basis evaluation is depicted in Figure 8.6. This information type contains relations which describe assessments of individual and collections of design requirements.

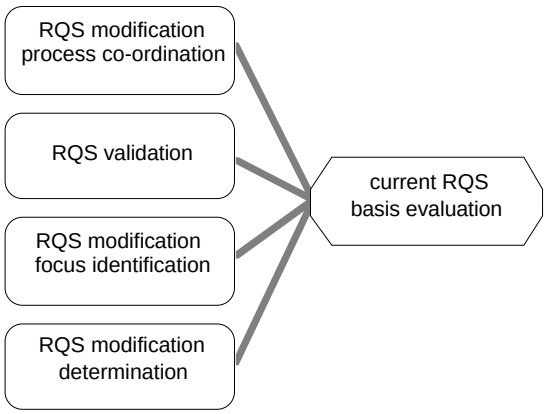


**Figure 8.6** Information type current RQS basis evaluation and sub-processes of RQS modification which use this information type.

The relations defined in the information type current RQS basis evaluation include:

**relations**
conflict_in_current_RQS;

The relation conflict in current RQS denotes that one or more conflicts have been found in the current RQS.

The information type RQS modification focus is depicted in Figure 8.7. This information types refers to three information types: possible RQS modification focus, selected RQS modification focus, and rejected RQS modification focus. The modification focus (on one RQS) consists of possible foci for modification, one or more selected foci for modification, and rejected foci for modification.

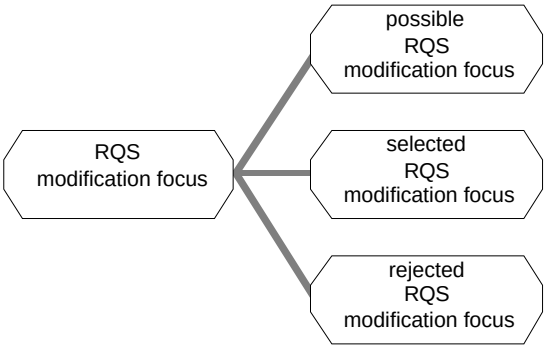


**Figure 8.7** Information type RQS modification focus.

The relations defined in the information types related to RQS modification focus are:

**relations**
qr_possible_as_focus,

qr_selected_as_focus,
qr_rejected_as_focus: qualified_requirement_name;

A focus on a qualified requirement implies that (unqualified) requirements (related to the qualified requirements in focus) are also focussed on.

The information type RQS modification strategy is depicted in Figure 8.8. This information types consists of three information types: RQS modification strategy specification, rejected RQS modification focus, and rejected RQS modification. The modification strategy is defined in RQS modification strategy specification, and rejected modification foci and rejected modifications to a RQS are described in Appendix B.

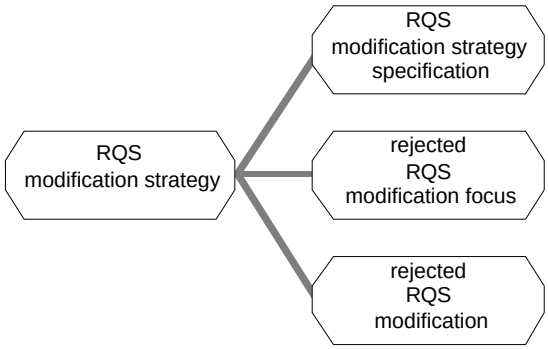


**Figure 8.8** Information type RQS modification strategy.

The relation defined in the information type RQS modification strategy specification is:

**relations**
modification_strategy: modification_method_identification *
modification_method_characterisation;

A modification strategy consists of a modification method identification and characterisation. The modification methods which can be identified include extension, restriction, and validation of design requirements. Modification method characterisation includes, for example, an indication of what needs to be restricted to (e.g., non knowledge composition design requirements).

Information types related to RQS modification determination are shown in Figure 8.9. The information type modification status contains information on the results of applying a modification method. The information type modification method contains information on the modification method to apply. The information types restriction results and extension results contain results of the restriction and extension methods.

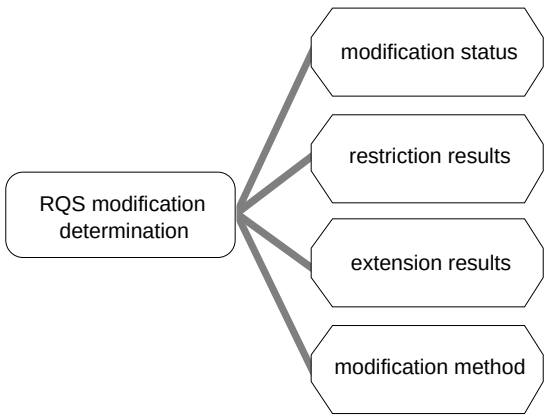


**Figure 8.9** Information types used in interfaces of sub-components of RQS modification determination.

Relations defined in the information type modification status are:

```
relations
    modification_finished;
    modification_consequences_needed;
```

These two relations indicate whether a modification step has been finished, or that information on consequences of the partial modification is needed in order to continue this particular modification step.

The relation defined in the information type modification method is:

```
relations
    selected_RQS_modification_method:    RQS_modification_method;
```

This relation is employed to identify the specific modification method to apply to the current RQS.

The relations defined in the information type restriction results are:

```
relations
    remaining_qr,
    qr_to_be_removed,
    req_to_be_removed:                   design_requirement_property_atom ;
```

The sort design requirement property atom contains the meta-description of the information type design requirement information. The relation remaining qr denotes which qualified requirements are to remain in the current RQS. The relations qr to be removed and req to be removed denote which design requirements are to be removed from the current RQS.

The relations defined in the information type extension results are:

```
relations
    new_qr_name:                         qualified_requirement_name ;
    new_qr_qualification:                qualified_requirement_name *
                                         qualification ;
    new_qr_associated_wff:               qualified_requirement_name *
                                         design_object_property_expression;
    new_refinement:                      qualified_requirement_name *
                                         qualified_requirement_name ;
```

The relations describe a new qualified requirement without mentioning a requirement. A qualified requirement can be related to a requirement expression without introducing new requirements: this is done to guarantee a minimal set of unique requirements and unique requirement expressions. The relation new qr name describes the name of a new qualified requirement. The relation new qr qualification associates a qualification to the new qualified requirement name. The relation new qr associated wff associates a requirement expression to the new qualified requirement name. The relation new refinement describes which qualified requirement is refined by which other qualified requirement.

A number of knowledge bases related to RQS modification are shown in Figure 8.10. They are described in some detail in the remainder of this section.

The knowledge base validation focus determination knowledge is employed to relate the contents of a RQS to foci for deductive refinement. Examples are shown below.

**Example from knowledge base validation focus determination knowledge**

The knowledge elements shown below all have the same format: the condition describes the presence of a qualified requirements in the current RQS. The conclusion describes which derivable design requirement relation is part of the focus for deductive refinement. The first knowledge element specifies that for each qualified requirement in the current RQS, it has to be determined whether the qualified requirement is part of a (i.e., one or more) conflict with other qualified requirements.

```
if holds(    is_qualified_requirement(    QR: qualified_requirement_name,
                                          Q: qualification,
                                          R: requirement_name ),
             pos )
```

```
then    is_part_of_RQS_refinement_focus(    qualified_requirement_in_a_conflict(
                                                QR: qualified_requirement_name),
                                            pos );
```

The second knowledge element specifies that for each qualified requirement in the current RQS, it has to be determined whether the qualified requirement has one or more other qualified requirements as refinements.

```
if  holds(      is_qualified_requirement(       QR: qualified_requirement_name,
                                                Q: qualification,
                                                R: requirement_name ),
                pos )
then    is_part_of_RQS_refinement_focus(    has_refinement(
                                                QR: qualified_requirement_name ),
                                            pos );
```

The third knowledge element specifies that for each qualified requirement in the current RQS, is has to be determined whether its structural influence is restricted to only the knowledge composition (of the system to be re-designed).

```
if  holds(      is_qualified_requirement(       QR: qualified_requirement_name,
                                                Q: qualification,
                                                R: requirement_name ),
                pos )
then    is_part_of_RQS_refinement_focus(    qr_structural_influence(
                                                QR: qualified_requirement_name,
                                                restricted_to_knowledge_composition ),
                                            pos );
```

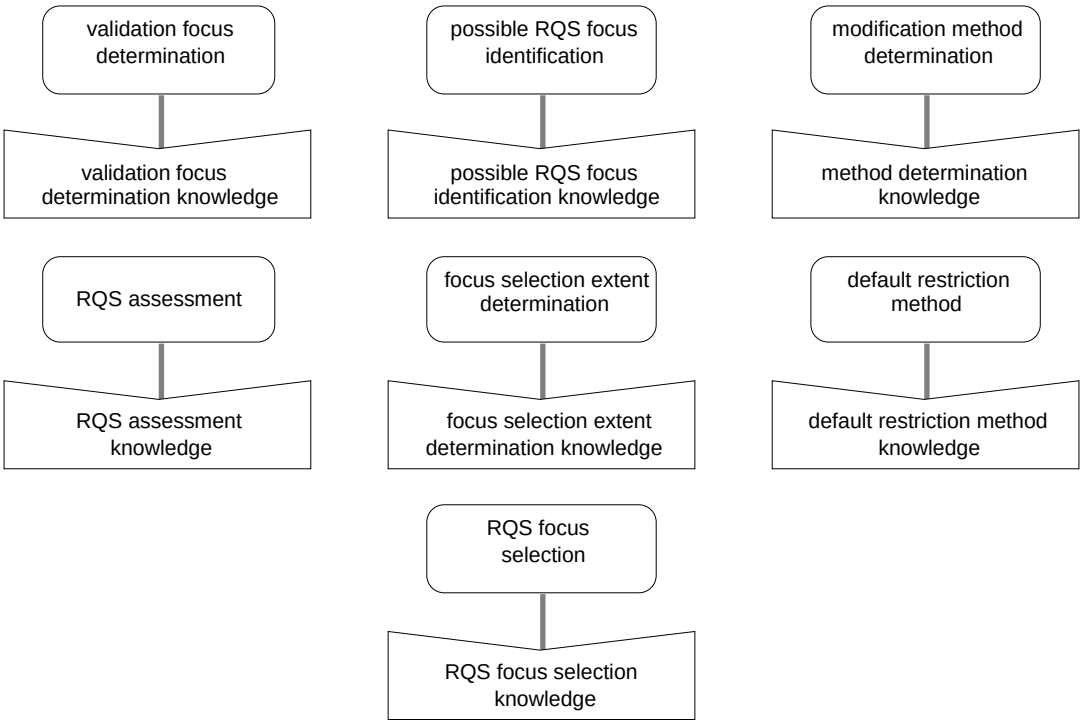


**Figure 8.10** Knowledge bases related to sub-components of RQS modification.

The knowledge base RQS assessment knowledge is employed to assess individual design requirements in the current RQS. An example is shown below.

**Example from knowledge base RQS assessment knowledge**

The knowledge element below specifies that if a qualified requirement is in a conflict, then the current RQS contains a conflict (which needs to be resolved later).

```
if   holds( qualified_requirement_in_a_conflict( QR: qualified_requirement_name ), pos )
then      conflict_in_current_RQS;
```

The knowledge base possible RQS focus identification knowledge is employed to identify possible foci for modification. An example is shown below.

**Example from knowledge base possible RQS focus identification knowledge**

The first condition in this knowledge element specifies that the modification strategy is to restrict the current RQS contents to those qualified requirements which do *not* have structural influence on the knowledge composition. The second condition specifies that a qualified requirement may not be rejected as a focus. The third condition specifies that the structural influence of this qualified requirement is not restricted to knowledge composition. The conclusion is that such a qualified requirement is a possible focus for modification.

```
if   modification_strategy(      restrict_to,
                                 non_knowledge_composition_design_requirements )
     and   not qr_rejected_as_focus(          QuRe: qualified_requirement_name )
     and   holds(      qr_structural_influence(     QuRe: qualified_requirement_name,
                                                    restricted_to_knowledge_composition ),
                       neg )
then      qr_possible_as_focus(     QuRe: qualified_requirement_name );
```

The knowledge base focus selection extent determination knowledge is employed to determine the number of foci which have to be selected from the possible foci. The knowledge base is described below.

**Contents of knowledge base focus selection extent determination knowledge**

The knowledge elements below specify a relation between which modification method is to be employed, and the number of modification foci which need to be selected: as many as possible, or one.

```
if   modification_strategy( restrict_to, S: modification_method_characterisation )
then      as_many_as_possible;

if   RQS_modification_strategy( extend, S: modification_method_characterisation )
then      one_at_a_time;
```

The knowledge base RQS focus selection knowledge is employed to select qualified requirements as modification foci, based on qualified requirements which are possible modification foci. The number of selected foci depends on task control settings for the extent of reasoning of the component RQS focus selection. An example from this knowledge base is shown below.

**Example from knowledge base RQS focus selection knowledge**

The knowledge element below species that for the modification method extend a qualified requirement which is a possible focus, and not rejected as a focus, can be selected as a modification focus.

```
if   modification_strategy(           extend,
                                      S: modification_method_characterisation )
     and   qr_possible_as_focus(      QR: qualified_requirement_name )
     and   not qr_rejected_as_focus(  QR: qualified_requirement_name )
then      qr_selected_as_focus(       QR: qualified_requirement_name );
```

The knowledge base modification method knowledge is employed to select a specific modification method (a sub-component of RQS modification determination) to modify the current RQS. The component corresponding to the selected modification method is activated by means of evaluation criteria and task control in the component RQS modification determination. An example from this knowledge base is shown below.

**Example from knowledge base modification method knowledge**

The knowledge element below specifies that if the modification strategy specifies that the current RQS has to be restricted, then the restriction method is selected as the modification method.

```
if   modification_strategy( restrict_to, C: modification_method_characterisation )
then      selected_RQS_modification_method( restriction_method );
```

The knowledge base restriction method knowledge is employed to specify which design requirements are to be removed from the current RQS, on the basis of a focus on those design requirements which have to *remain* in the current RQS. Examples from this knowledge base are described below.

**Example from knowledge base restriction method knowledge**

The knowledge element below specifies that the qualified requirement that has *not* been selected as a modification focus is to be removed from the current RQS.

```
if  not   qr_selected_as_focus(          QR: qualified_requirement_name )
    and   holds( is_qualified_requirement( QR: qualified_requirement_name,
                                           Q: qualification,
                                           R: requirement_name ),
              pos )
then      qr_to_be_removed( is_qualified_requirement(  QR: qualified_requirement_name,
                                                       Q: qualification,
                                                       R: requirement_name ) );
```

## 8.4 Refinement of RQSM history maintenance

Both the process composition and the knowledge composition for the process RQSM history maintenance are described below.

### 8.4.1 Process composition for RQSM history maintenance: identification of processes and abstraction levels

The process of RQSM History Maintenance is responsible for maintaining and retrieving information on the RQSM process for future use. In particular, information on the sets of qualified requirements considered (accepted and/or rejected) and information on states within the RQS modification process are stored. A number of sub-processes are performed to this purpose, as shown in Figure 8.11. The process RQSM history maintenance is composed of the process RQS history maintenance and the process RQS modification state history maintenance.

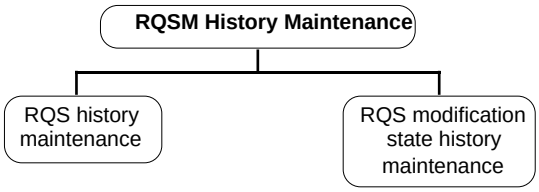


**Figure 8.11** Processes at different abstraction levels for RQSM History Maintenance.

The process RQS history maintenance is responsible for storing, retrieving and managing sets of design requirements over time. The process RQS modification state history maintenance is responsible for storing, retrieving and managing modification states (i.e. information regarding the modification process) over time. These processes are further described in Appendix B.1.3.

The *interface information types* for the two sub-processes of RQSM History Maintenance are listed in Table 8.5 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| RQS history maintenance | • RQS assessment<br>• RQS assessment history queries<br>• DOD assessment<br>• DOD assessment history queries<br>• current RQS contents<br>• RQS history queries<br>• current RQS replacement request<br>• RQS | • RQS assessment history search results<br>• DOD assessment history search results<br>• RQS history search results<br>• new current RQS contents<br>• current RQS name<br>• current RQS replacement results |
| RQS modification state history maintenance | • given current RQS name<br>• RQS modification progress<br>• RQS modification state history queries<br>• overall design strategy | • RQS modification state history search results |

**Table 8.5** Interface information types for sub-processes of RQSM History Maintenance.

- The process RQS history maintenance requires information on the assessment of sets of qualified requirements (RQS assessment), queries on the RQS assessment history (RQS assessment history queries), assessment of design object descriptions (DOD assessment), queries on the DOD assessment history (DOD assessment history queries), contents of the current RQS (current RQS contents), queries on the RQS history (RQS history queries), and requests for replacement of the current RQS (current RQS replacement request). This process produces results of searching the RQS assessment history (RQS assessment history search results), results of searching the DOD assessment history (DOD assessment history search results), results of searching the RQS history (RQS history search results), the contents of the new, current, RQS (new current RQS contents), the name of the current RQS stored in RQS history (current RQS name), and results on the success of replacing the current RQS (current RQS replacement results).
- The process RQS modification state history maintenance needs information on the given name of the current RQS stored in RQS history (given current RQS name), progress of the modification process (RQS modification progress), overall design strategy (overall design strategy), and queries on the RQS modification state history (RQS modification state history queries). This process generates results of searching the RQS modification state history (RQS modification state history search results).

### 8.4.2 Process composition relation within RQSM history maintenance

The information links in the component RQSM History Maintenance are shown in Figure 8.12.

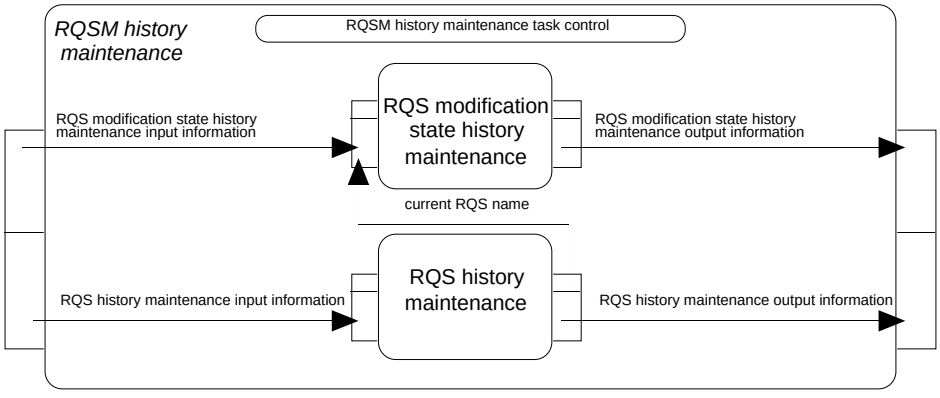


**Figure 8.12** Information links within the process of RQSM history maintenance.

Within this component four mediating links and one private link are defined:

- The mediating link RQS modification state history maintenance input information transfers information expressed in RQS modification progress, RQS information state history queries, and overall design strategy from the input interface of RQSM history maintenance to the input interface of RQS modification state history maintenance.
- The mediating link RQS history maintenance input information transfers information expressed in RQS assessment, RQS assessment history queries, DOD assessment, DOD assessment history queries, current RQS contents, RQS history queries, and current RQS replacement request from the input interface of RQSM history maintenance to the input interface of RQS history maintenance.
- The private link current RQS name transfers the name of the RQS currently being stored (current RQS name) from the output interface of RQS history maintenance to the given name of the RQS currently being stored (given current RQS name) in the input interface of RQS modification state history maintenance on the basis of an explicit mapping between these information types.
- The mediating link RQS modification state history maintenance output information transfers information expressed in RQS modification state history search results from the output interface of RQS modification state history maintenance to the output interface of RQSM history maintenance.
- The mediating link RQS history maintenance output information transfers information expressed in RQS assessment history search results, DOD assessment history search results, RQS history search results, new current RQS contents, current RQS replacement results and current RQS name from the output interface of RQS history maintenance to the output interface of RQSM history maintenance.

The task control within the component RQSM history maintenance is as follows. Upon activation of RQSM history maintenance a number of situations are possible, which correspond to task control foci for this component. The task control foci are: initial storage of information, continued storage of information, update of the history, execute queries, and replacement of current RQS preparation. The first four task control foci result in the activation of all components and information links, the task control focus replacement of current RQS preparation results in a slightly different activation of components and information links:

- On activation with one of the task control foci initial storage of information, continued storage of information, update of the history, and execute queries the information links RQS modification state history maintenance input information and RQS history maintenance input information are made up-to-date, and RQS history maintenance is activated with the same task control focus as RQSM history maintenance. Upon termination of RQS history maintenance, RQS modification state history maintenance is activated with the same task control focus as RQSM history maintenance and the information link current RQS name is made up-to-date. Upon termination of RQS modification state history maintenance, the information links RQS modification state history maintenance output information and RQS history maintenance output information are made up-to-date and RQSM history maintenance terminates itself.
- On activation with the task control focus replacement of current RQS preparation, the information link current RQS history maintenance input information is made update, and RQS history maintenance is activated with task control focus ‘replacement of current RQS preparation’. Upon termination of RQS history maintenance the information link RQS history maintenance output information is made up-to-date and RQSM history maintenance terminates itself.

### 8.4.3 Knowledge composition for RQSM history maintenance

Most of the information types and knowledge bases related to the process RQSM history maintenance are not instantiated, one exception is the information type RQS modification state information, as shown in Figure 8.13.

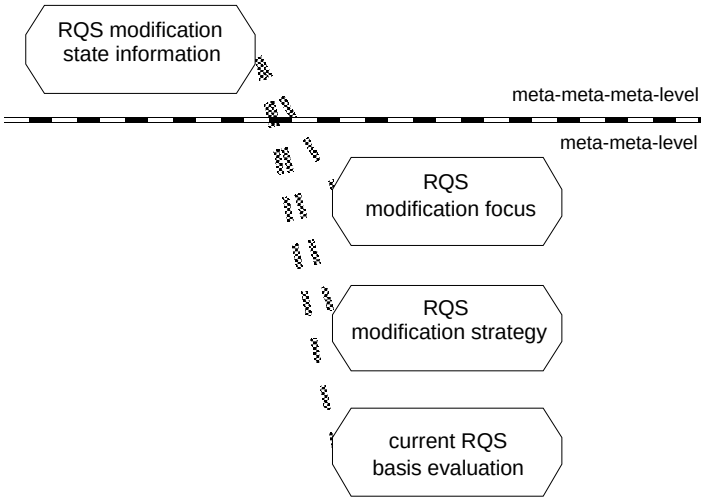


**Figure 8.13** Specialisation of the information type RQS modification state information.

The meta-descriptions of the information types RQS modification focus, RQS modification strategy, and current RQS basis evaluation are related to the sort RQS modification state attribute value (see Section A.3.3), thereby specialising the relation has RQS modification state value.

## 8.5 Summary

In this chapter the results of the previous chapter are extended: the generic model for design is further refined. The refinement of the process RQS manipulation is addressed in this chapter. Four refinement steps have been described, as is depicted in Figure 8.14.

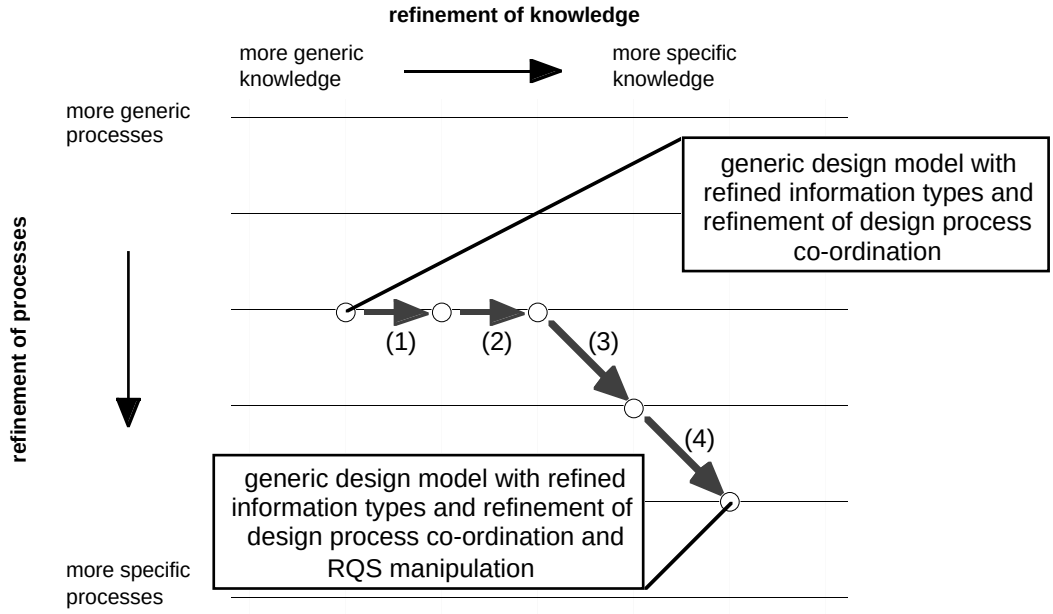


**Figure 8.14** Refinement steps described in this chapter: the numbered arrows correspond to Sections 8.1 to 8.4.

In the first refinement step the information types and knowledge-base related to the process current RQS maintenance are instantiated. In the second refinement step the information types and knowledge base related to the process deductive RQS refinement are instantiated. In the third refinement step the process RQS modification is partially refined; additional refinements of sub-processes of RQS modification are described in Appendix B.1.1 and B.1.2. In the fourth

refinement step the process RQSM history maintenance is partially refined; additional refinements of sub-processes of RQSM history maintenance are described in Appendix B.1.3.

The description of the refinement of the process RQS manipulation is the realisation of the desideratum dr4.

The refinement of the process RQS manipulation is more detailed than the initial description of the structure of RQS modification (and DOD modification), described in (Brazier, Langen, Treur and Wijngaards, 1996), see Figure 8.15, an earlier refinement of part of the generic model of design.

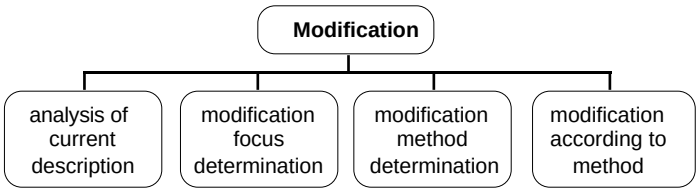


**Figure 8.15** Composition of modification sub-components of RQS modification (and DOD modification) from (Brazier, Langen, Treur, Wijngaards, 1996).

The processes depicted in Figure 8.15 are refinements of the component RQS modification of the generic task model of design:

- analysis of current description, that investigates which conflicts, unsatisfied design requirements, etc., are in the current RQS,
- modification focus determination, that determines which parts of the current RQS must be modified to be able to resolve the identified problems,
- modification method determination, that determines the method for modifying the parts of the current RQS in focus,
- modification according to method, that modifies the parts of the current RQS in focus, according to the method determined.

The processes distinguished in Figure 8.15 can be found in the compositions of RQS modification (Sections 8.3). The analysis of current description is located in RQS validation; modification focus determination is located within RQS modification focus identification; modification method determination is located within RQS modification determination; and modification according to method is realised by task control knowledge and several processes corresponding to methods in RQS modification determination.

The process composition and knowledge composition described in this chapter are generic in the sense that more elaborate, knowledge-intensive specialisations can be added. For example, in the refinement for RQS modification determination additional modification methods can be included.

# 9 Design Object Description Manipulation in the Re-design Model for Compositional Systems

The generic model for design has been (partially) refined, in the two previous chapters. In this chapter additional refinements are described for the DOD Manipulation process. The realisation of the following desideratum is addressed in this chapter:

- dr3, model of the manipulation of compositional system structures,

The process design object description manipulation (DODM) modifies design object descriptions on the basis of design strategies from DPC and design requirements from RQSM. The process of DODM is composed of four sub-processes (see Figure 6.9): DODM history maintenance, DOD modification, current DOD Maintenance, and deductive DOD refinement. The first two sub-processes are composed, the last two are not. Section 9.1 describes a refinement of current DOD maintenance, Section 9.2 describes a refinement of deductive DOD refinement, Section 9.3 describes a refinement of DOD modification, and Section 9.4 describes a refinement of DODM history maintenance. In each of these sections both process composition and knowledge composition is addressed. In Section 9.5 the refinement of the process DOD manipulation is briefly discussed.

## 9.1 Refinement of current DOD maintenance

The process current DOD maintenance is not composed. The information types in the interface of current DOD maintenance, discussed in Section 6.1.2, are used as described in Section 7.1. Knowledge employed in this process is instantiated for the domain of re-design of compositional systems. The process current DOD maintenance employs knowledge to, e.g., distinguish viewpoints: a focus can be made, highlighting part of a compositional system corresponding to a particular viewpoint. An example of an instance of knowledge used to focus on part of a compositional system is given below.

**Example from knowledge base current DOD maintenance knowledge**

The knowledge element below has, as its first condition, the current component in focus. The second condition identifies sub-components of component C. The conclusion states that sub-components of component C are in the current focus.

```
if   current_focus(                 subcomponents_of,
                                    C: component_name )
     and   has_subcomponent(        C: component_name,
                                    D: component_name )
then       in_current_focus(        D: component_name );
```

## 9.2 Refinement of deductive DOD refinement

The process deductive DOD refinement is not composed. The information types in the interface of deductive DOD refinement, discussed in Section 6.1.2, are used as described in Section 7.1. Knowledge employed in this process is instantiated for the domain of re-design of compositional systems. The process deductive DOD refinement can employ knowledge to, e.g., derive properties of compositional systems: properties which may be required. The deductive

knowledge relating the structure of a compositional system to properties of a compositional system (described in Chapter 5) is used in this process. Example knowledge to deduce the property of is capable of bi-directional communication is given below (for other instances, see Section 5.4).

**Example from knowledge base deductive DOD refinement knowledge**

The knowledge shown specifies the deduction of the properties related to the property is capable of bi-directional communication. The first knowledge element specifies that if a component C has the properties is capable of reasoning about unidirectional communication from a component D, is capable of executing unidirectional communication from component D, and combines reasoning about and executing unidirectional communication component D, then component C has the property is capable of unidirectional communication from component D.

```
if  has_property(                C: Component_name,
                                 is_capable_of_reasoning_about_unidirectional_
                                       communication_from( D: Component_name ) )
    and   has_property(          C: Component_name,
                                 is_capable_of_executing_unidirectional_
                                       communication_from( D: Component_name ) )
    and   has_property(          C: Component_name,
                                 combines_reasoning_about_and_executing_unidirectional_
                                       communication_from( D: Component_name ) )
    then  has_property(          C: Component_name,
                                 is_capable_of_unidirectional_communication_from(
                                       D: Component_name ) );
```

The second knowledge element specifies that if a component C is characterised as an agent, and component C has an input interface information type Cin which is characterised as used for unidirectional communication from an agent, and component C has a sub-component D which is characterised as an agent interaction management component, and component D has an input interface type Din which is characterised as used for unidirectional communication from an agent, and this component D has a knowledge base K which is characterised as knowledge on unidirectional communication from an agent, and the information types Cin and Din, and the knowledge base K are specialised for unidirectional communication from component C2 which is also an agent, then component C has the property is capable of reasoning about unidirectional communication from component C2.

```
if  is_component(                      C: component_name )
    and   has_characterisation(        C: component_name,
                                       agent )
    and   has_interface_information_type( C: component_name,
                                          input_interface,
                                          Cin: information_type_name )
    and   has_characterisation(        Cin: information_type_name,
                                       unidirectional_communication_from )
    and   has_subcomponent(            C: component_name,
                                       D: component_name )
    and   has_characterisation(        D: component_name,
                                       agent_interaction_management_component )
    and   has_interface_information_type( D: component_name,
                                          input_interface,
                                          Din: information_type_name )
    and   has_characterisation(        Din: information_type_name,
                                       unidirectional_communication_from )
    and   has_knowledge_base(          D: component_name,
                                       K: knowledge_base_name )
    and   has_characterisation(        K: knowledge_base_name,
                                       unidirectional_communication_from_knowledge )
    and   is_information_link(         I: information_link_name )
    and   has_information_link(        C: component_name,
                                       I: information_link_name )
    and   has_source_component(        I: information_link_name,
                                       C: component_name )
```

```
    and   has_destination_component(          I: information_link_name,
                                              D: component_name )
    and   has_source_information_type(        I: information_link_name,
                                              Cin: information_type_name )
    and   has_destination_information_type(   I: information_link_name,
                                              Din: information_type_name )
    and   is_component(           C2: component_name )
    and   has_characterisation(   C2: component_name,
                                  agent )
    and   has_characterisation(   K: knowledge_base_name,
                                  specialised_for( C2: component_name ) )
    and   has_characterisation(   Cin: information_type_name,
                                  specialised_for( C2: component_name ) )
    and   has_characterisation(   Din: information_type_name,
                                  specialised_for( C2: component_name ) )
then      has_property(           C: component_name,
                                  is_capable_of_reasoning_about_unidirectional_
                                        communication_from( C2: component_name ) );
```

## 9.3 Refinement of DOD Modification

For the process DOD modification first the process composition is described, then the knowledge composition.

### 9.3.1 Process composition of DOD modification: identification of processes and abstraction levels

The process DOD modification determines modifications to a design object description (DOD) to construct a DOD which adheres to the design requirements given to DODM. To this purpose a number of sub-processes are distinguished as shown in Figure 9.1. The process DOD modification process co-ordination is responsible for the co-ordination of the entire process within DODM: this process determines whether, when, and by which means, a particular DOD is to be modified.

The global phases within DOD modification resemble a process control model (e.g., controlling a chemical process). In a process control task a cycle occurs over the sub-tasks: analysis, planning, execution. Similarly, within DOD modification analysis is performed by DOD validation, planning is performed by DOD modification focus identification and DOD modification determination, and execution is performed by effectuating modifications to a DOD, resulting in a new DOD in current DOD maintenance.

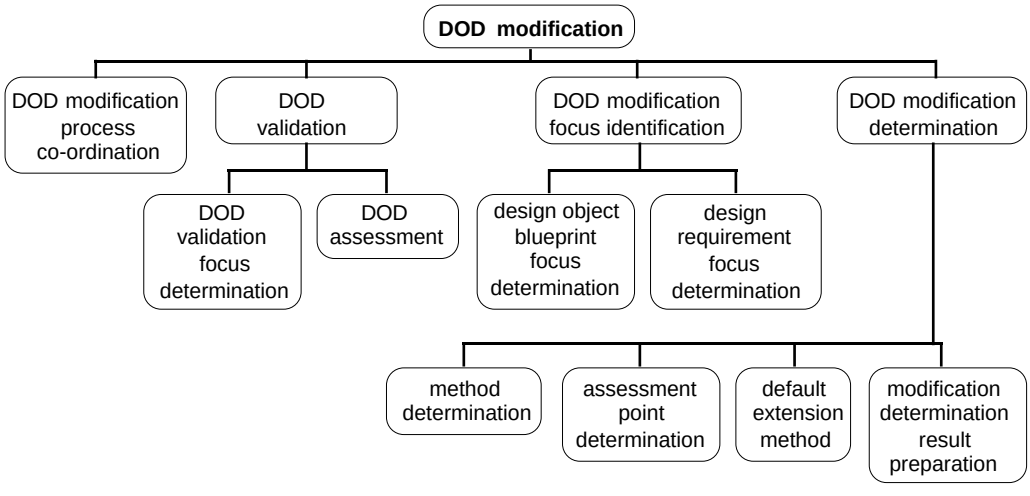


**Figure 9.1** Partial process refinement for DOD modification.

The process DOD modification process co-ordination co-ordinates the modification process on the basis of the overall design strategy and information available in the DODM history. A modification strategy is issued, which influences the other three sub-processes of DOD modification. On the basis of results obtained from these three sub-processes, revised modification strategies can be issued, histories can be inspected, information can be stored in the history, or the DOD modification process can be terminated. This process is described in more detail in Appendix B.2.1.

The process DOD validation validates the current DOD. Its sub-process validation focus determination determines which properties of design object descriptions need to be validated on the basis of the current design requirements. DOD refinement (see Figure 6.9) is given these results as goals to pursue. The sub-process DOD assessment assesses the achievement of the validation focus on the basis of the results of the refinement.

The process DOD modification focus identification determines which part of the design object description needs to be modified, to satisfy which particular design requirement, according to the given internal strategy. Two sub-processes are distinguished: design requirement focus determination identifies a design requirements which needs to be satisfied, and design object blueprint focus determination identifies part of the design object description related to the possible satisfaction of the design requirements in focus.

The process DOD modification determination determines the actual modifications to the current DOD on the basis of given strategies from DOD modification process co-ordination, and information on the modification focus from DOD modification focus identification. This process entails four sub-processes. The process method determination chooses the best method corresponding to the given strategy, given the current information. The process assessment point determination determines points of interest related to the modification focus and the current DOD on the basis of assessments of design requirements. On the basis of these assessment points the process default extension method determines modifications to the current DOD. The process modification determination result preparation analyses these suggested modification, e.g., to ascertain non-ambiguous names of parts of design object descriptions, before formulating the final modifications to the current DOD. The process assessment point determination is described in more detail in Appendix B.2.2.

The *interface information types* of the sub-processes of DOD modification are listed in Table 9.1 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| DOD modification process co-ordination | • DOD modification state history search results<br>• overall design strategy<br>• DOD assessment history search results<br>• RQS history search results<br>• DOD history search results<br>• current DOD replacement results<br>• DOD modification focus<br>• current DOD modification<br>• current DOD basis evaluation | • DOD modification progress<br>• current manipulation action<br>• DOD modification state history queries<br>• DOD assessment<br>• DOD assessment history queries<br>• RQS history queries<br>• DOD history queries<br>• current DOD replacement request<br>• DOD modification strategy |
| DOD Validation | • current DOD contents<br>• current design requirements | • DOD refinement goals<br>• current DOD basis evaluation |
| DOD Modification Focus Identification | • DOD modification strategy<br>• current DOD basis evaluation<br>• current DOD contents<br>• current design requirements | • DOD modification focus |
| DOD Modification Determination | • DOD modification strategy<br>• current DOD basis evaluation<br>• current design requirements<br>• DOD modification focus<br>• current DOD contents | • modification status<br>• current DOD modification |

**Table 9.1** Interface information types for sub-processes of DOD modification.

- The process DOD modification process co-ordination needs results of searching the DOD modification state history ( DOD modification state history search results), an overall design strategy (overall design strategy), results of searching the DOD assessment history (DOD assessment history search results), results of searching the sets of qualified requirements history (RQS history search results), results of searching the DOD history (DOD history search results), results on the success of replacing the current DOD (current DOD replacement results), assessment of the current DOD on the basis of design requirements (current DOD basis evaluation), modification foci (DOD modification focus), and modifications to the current DOD (current DOD modifications). This process generates information on the progress of the modification process (DOD modification progress), actions for the manipulation process (current manipulation action), queries on DOD modification state history ( DOD modification state history queries), assessment of design object descriptions (DOD assessment), queries on the DOD assessment history (DOD assessment history queries), queries on RQS history (RQS history queries), queries on DOD history (DOD history queries), requests for replacement of the current DOD (current DOD replacement request), and a modification strategy (DOD modification strategy).
- The process DOD validation requires the contents of the current DOD (current RQS contents) and design requirements (current design requirements), and produces goals to achieve for deductive refinement of a DOD (DOD refinement goals) and the assessment of the current DOD on the basis of design requirements (current DOD basis evaluation).
- The process DOD modification focus identification requires a modification strategy (DOD modification strategy), assessment of the current DOD on the basis of design requirements (current DOD basis evaluation), the contents of the current DOD (current DOD contents), and design requirements (current design requirements). The process DOD modification focus identification generates modification foci (DOD modification focus).
- The process DOD modification determination requires the modification strategy (DOD modification strategy), assessment of the current DOD on the basis of design requirements (current DOD basis evaluation), design requirements (current design requirements), modification foci (DOD modification focus), and the contents of the current DOD (current DOD contents). The results of this process are an indication of the status of the modification process (modification status), and modifications to be performed on the current DOD (current DOD modifications).

| *process* | *input information type* | *output information type* |
|---|---|---|
| validation focus determination | • current DOD contents<br>• current design requirements | • DOD refinement goals |
| DOD assessment | • current DOD contents<br>• DOD refinement goals<br>• current design requirements | • current DOD basis evaluation |

**Table 9.2** Interface information types for sub-processes of DOD validation.

The interface information types of the sub-processes of DOD validation are listed in Table 9.2 and described below.

- The process validation focus determination requires information on the contents of the current DOD (current DOD contents) and design requirements (current design requirements), and produces information on goals to achieve for deductive refinement of a DOD (DOD refinement goals).
- The process assessment requires information on the contents of the current DOD (current DOD contents), design requirements (current design requirements), and goals to achieve for deductive refinement of a DOD (DOD refinement goals). This process produces information on the assessment of the current DOD on the basis of design requirements (current DOD basis evaluation).

| *process* | *input information type* | *output information type* |
|---|---|---|
| design requirement focus determination | • DOD modification strategy<br>• current DOD contents<br>• current design requirements<br>• current DOD basis evaluation | • design requirement focus<br>• intermediate blueprint focus |
| design object blueprint focus determination | • DOD modification strategy<br>• current DOD contents<br>• intermediate blueprint focus | • design object blueprint focus |

**Table 9.3** Interface information types for sub-processes of DOD modification focus identification.

The interface information types of the sub-processes of DOD modification focus identification are listed in Table 9.3 and described below.

- The process design requirement focus determination requires a modification strategy (DOD modification strategy), the contents of the current DOD (current DOD contents), design requirements (current design requirements), and an assessment of the current DOD on the basis of design requirements (current DOD basis evaluation). It produces a focus on design requirements (design requirement focus) and an intermediate focus on the structure of a DOD (intermediate blueprint focus).
- The process design object blueprint focus determination needs a modification strategy (DOD modification strategy), the contents of the current DOD (current DOD contents) and an intermediate focus on the structure of a DOD (intermediate blueprint focus). It produces a focus on the structure of a DOD (design object blueprint focus).

| *process* | *input information type* | *output information type* |
|---|---|---|
| method determination | • DOD modification strategy<br>• current DOD basis evaluation<br>• DOD modification focus<br>• modification method evaluation | • modification method<br>• modification status |
| assessment point determination | • modification method<br>• current DOD basis evaluation<br>• DOD modification focus<br>• current DOD contents<br>• current design requirements | • modification method evaluation<br>• assessment point to be realised |
| default extension method | • modification method<br>• current DOD contents<br>• assessment point to be realised<br>• DOD modification focus<br>• rejected modifications | • extension results |
| modification determination result preparation | • extension results<br>• current DOD contents | • selected DOD modification |

**Table 9.4** Interface information types for sub-processes of DOD modification determination.

The interface information types of the sub-processes of DOD modification determination are listed in Table 9.4 and described below.

- The process method determination requires a modification strategy (RQS modification strategy), assessment of the current DOD on the basis of design requirements (current DOD basis evaluation), and modification foci (DOD modification focus). This process generates a method for modification (modification method) and a status of this modification process (modification status).
- The process assessment point determination needs requires a method for modification (modification method), an assessment of the current DOD on the basis of design requirements (current DOD basis evaluation), a focus for modification (DOD modification focus), the contents of the current DOD (current DOD contents), and design requirements (current design requirements). Its results are an evaluation of the method for modification

(modification method evaluation), and assessment points which need to be realised (assessment point to be realised).

- The process default extension method requires a method for modification (modification method), the contents of the current DOD (current DOD contents), assessment points which need to be realised (assessment point to be realised), a focus for modification (DOD modification focus), and rejected modifications (rejected modifications). This process produces extensions to the current DOD (extension results).
- The process modification determination result preparation needs the contents of the current DOD (current DOD contents), and extensions to the current DOD (extension results). Its results are selected modifications on the current DOD (selected DOD modification).

### 9.3.2 Process composition relation within DOD Modification

The information links in the component DOD modification are shown in Figure 9.2.

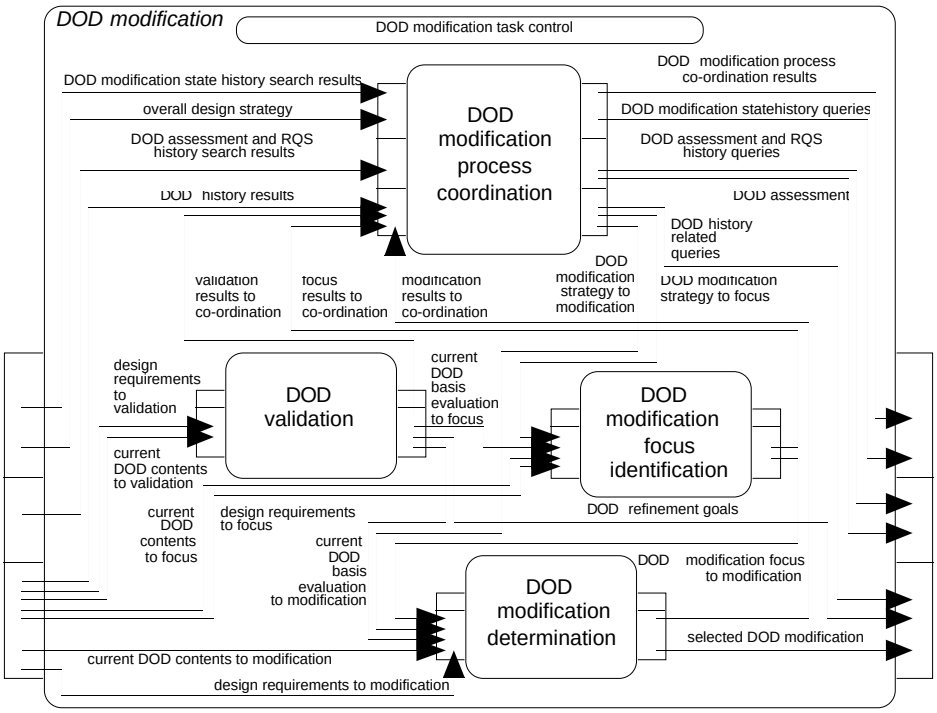


**Figure 9.2** Information links within the process of DOD modification.

Within this component seventeen mediating links and eight private links are defined:

- The mediating links current DOD contents to validation, current DOD contents to focus, and current DOD contents to modification transfer the contents of the current DOD (current DOD contents) from the input interface of DOD modification to the input interfaces of DOD validation, DOD modification focus identification, and DOD modification determination, respectively.
- The mediating links design requirements to validation, design requirements to focus, and design requirements to modification transfer the current design requirements (current design requirements) from the input interface of DOD modification to the input interfaces of DOD validation, DOD modification focus identification, and DOD modification determination, respectively.
- The mediating links DOD history results, DOD assessment and RQS history results, overall design strategy, and DOD modification state history search results transfer information expressed in DOD history search results and current DOD history search results, DOD assessment history search results and RQS history search results, overall design strategy, and

DOD modification state history search results from the input interface of DOD modification to the input interface of DOD modification process co-ordination, respectively.

- The private links current DOD basis evaluation to focus and current DOD basis evaluation to modification transfer an assessment of the current DOD on the basis of design requirements (current DOD basis evaluation) from the output interface of DOD validation to the input interfaces of DOD modification focus identification and DOD modification determination, respectively.
- The private link DOD modification focus to modification transfers foci for modification (DOD modification focus) from the output interface of DOD modification focus identification to the input interface of RQS modification determination.
- The private links validation results to co-operation, focus results to co-operation, and modification results to co-operation transfer results from the output interface of DOD validation, DOD modification focus identification, and DOD modification determination to the input interface of DOD modification process co-ordination.
- The private links DOD modification strategy to focus and DOD modification strategy to modification transfer the modification strategy (DOD modification strategy) from the output interface of DOD modification process co-ordination to the input interface of DOD modification focus identification and DOD modification determination, respectively.
- The mediating link selected DOD modification transfers modifications to be performed (selected DOD modification) from the output interface of DOD modification determination to the output interface of DOD modification.
- The mediating link DOD refinement goals transfers goals for deductive refinement of the contents of a DOD (DOD refinement goals) from the output interface of DOD validation to the output interface of DOD modification.
- The mediating links DOD history related queries, DOD assessment and RQS history queries, DOD assessment, DOD modification state history queries, and DOD modification process co-ordination results transfer information expressed in DOD history queries and current DOD replacement request, DOD assessment history queries and RQS history queries, DOD assessment, DOD modification state history queries, and DOD modification progress and current manipulation action from the output interface of DOD modification process co-ordination to the output interface of DOD modification, respectively.

The task control within the component DOD modification distinguishes a number of phases, which correspond to manipulation actions determined by DOD modification process co-ordination. Upon activation of DOD modification, DOD modification process co-ordination is activated. A distinction can be made between continuation of the previous manipulation process, or initiation of a new manipulation process. Queries on history, requests for replacement of the current DOD, and information on the modification process can be produced by DOD modification process co-ordination. DOD validation can be activated to determine refinement goals, and to assess the satisfaction of design requirements in relation to the current DOD. The modification strategy, also produced by DOD modification process co-ordination can be made available to both DOD modification focus identification and DOD modification determination, activated in that order. Selected modifications to the current DOD can be made available as output of DOD modification. A manipulation action determined by DOD modification process co-ordination indicates that DOD modification can terminate itself.

### 9.3.3 Process composition relation within DOD validation

The information links in the component DOD validation are shown in Figure 9.3. Within this component six mediating links and one private link are defined:

- The mediating link current DOD contents to focus transfers the contents of the current DOD (current DOD contents) from the input interface of DOD validation to the input interface of validation focus determination.

- The mediating link current design requirements to focus transfers the current design requirements (current design requirements) from the input interface of DOD validation to the input interface of validation focus determination.
- The mediating link current DOD contents to assessment transfers the contents of the current DOD (current DOD contents) from the input interface of DOD validation to the input interface of assessment.
- The mediating link current design requirements to assessment transfers the current design requirements (current design requirements) from the input interface of DOD validation to the input interface of assessment.
- The private link pessimistic assumption on goals transfers the occurrence of goals for deductive refinement of a DOD (epistemic of DOD refinement goals) from the output interface of validation focus determination to assumptions on not having achieved these goals (assumption on current DOD contents) in the input interface of assessment on the basis of an explicit mapping between these information types.
- The mediating link DOD refinement goals transfers goals for deductive refinement of the contents of a DOD (DOD refinement goals) from the output interface of validation focus determination to the output interface of DOD validation.
- The mediating link current DOD basis evaluation transfers the assessment of the current DOD on the basis of design requirements (current DOD basis evaluation) from the output interface of assessment to the output interface of DOD validation.

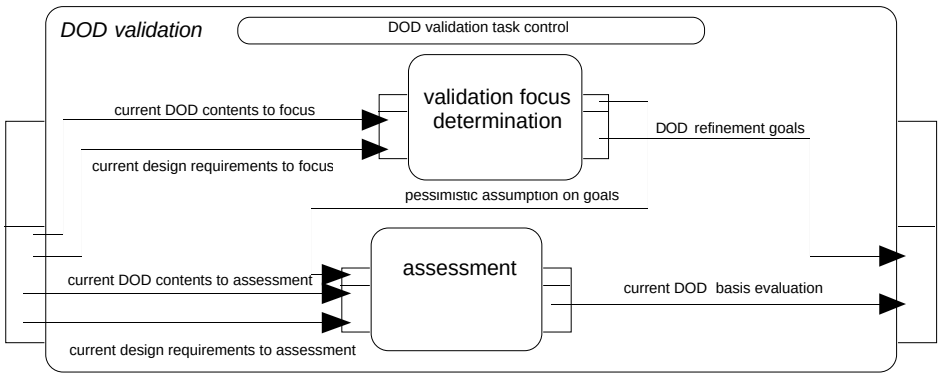


**Figure 9.3** Information links within the process of DOD validation.

The task control within the component DOD validation is as follows. Upon activation of DOD validation a number of possible situations are possible, which correspond to task control foci for this component. The task control foci are: prepare for validation, and assess validation results. For each of these task control foci specific links need to be made up-to-date:

- On activation with the task control focus prepare for validation the information links current DOD contents to focus, and current design requirements to focus are made up-to-date and validation focus determination is activated. Upon termination of validation focus determination, the information link DOD refinement goals is made up-to-date and DOD validation terminates itself.
- On activation with the task control focus assess validation results the information links current design requirement to assessment, current DOD contents to assessment, and pessimistic assumption on goals are made up-to-date and assessment is activated. Upon termination of assessment, the information link current DOD basis evaluation is made up-to-date and DOD validation terminates itself.

### 9.3.4 Process composition relation within DOD modification focus identification

The information links in the component DOD modification focus identification are shown in Figure 9.4.

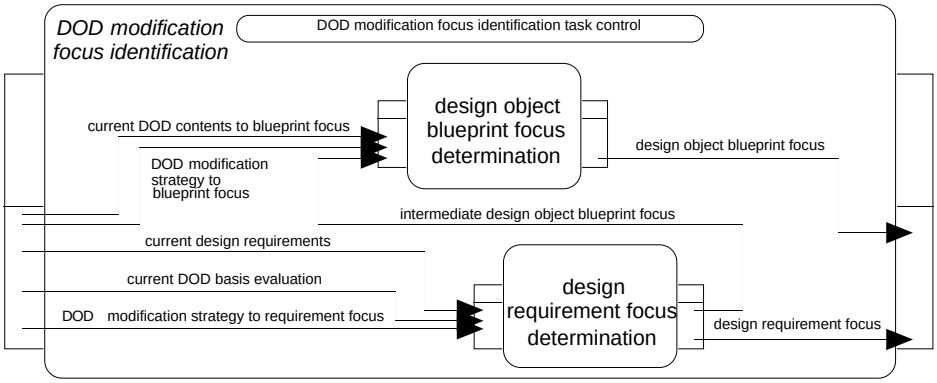


**Figure 9.4** Information links within the process of DOD modification focus identification.

Within this component seven mediating links and one private link are defined:

- The mediating link DOD modification strategy to requirement focus transfers the modification strategy (DOD modification strategy) from the input interface of DOD modification focus identification to the input interface of design requirement focus determination.
- The mediating link current design requirements transfers design requirements (current design requirements) from the input interface of DOD modification focus identification to the input interface of design requirement focus determination.
- The mediating link current DOD basis evaluation transfers an assessment of the current DOD on the basis of design requirements (current DOD basis evaluation) from the input interface of DOD modification focus identification to the input interface of design requirement focus determination.
- The mediating link DOD modification strategy to blueprint focus transfers the modification strategy (DOD modification strategy) from the input interface of DOD modification focus identification to the input interface of design object blueprint focus determination.
- The mediating link current DOD contents to blueprint focus transfers the contents of the current DOD (current DOD contents) from the input interface of DOD modification focus identification to the input interface of design object blueprint focus determination.
- The private link intermediate design object blueprint focus transfers an intermediate focus on the structure of the design object (intermediate blueprint focus) from the output interface of design requirement focus determination to the input interface of design object blueprint focus determination.
- The mediating link design object blueprint focus transfers selected foci for modification (design object blueprint focus) from the output interface of design object blueprint focus determination to the output interface of DOD modification focus identification.
- The mediating link design requirement focus transfers selected foci on design requirements (design requirement focus) from the output interface of design requirement focus determination to the output interface of DOD modification focus identification.

The task control within the component DOD modification focus identification is as follows. Upon activation of DOD modification focus identification, the information links current design requirements, current DOD basis evaluation, and DOD modification strategy to requirement focus are made up-to-date and design requirement focus determination is activated. Upon termination of design requirement focus determination the information links intermediate design object blueprint focus, DOD modification strategy to blueprint focus, and current DOD contents to blueprint focus are made up-to-date and design object blueprint focus determination is activated. Upon termination of design object blueprint focus determination, the information links design object blueprint focus and design requirement focus are made up-to-date and DOD modification focus identification terminates itself.

### 9.3.5 Process composition relation within DOD modification determination

The information links in the component DOD modification determination are shown in Figure 9.5.

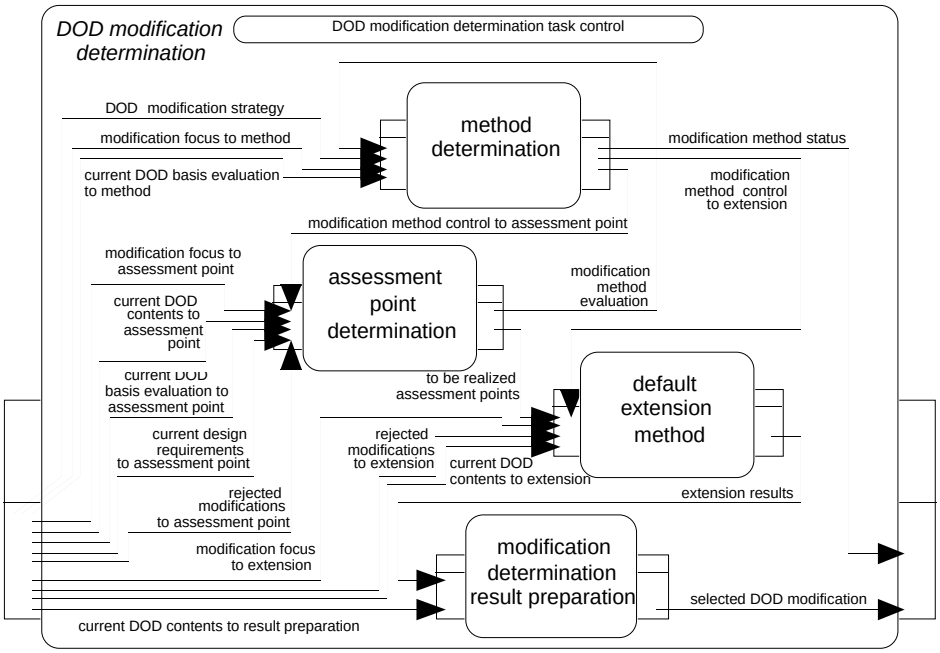


**Figure 9.5** Information links within the process of DOD modification determination.

Within this component fourteen mediating links and four private links are defined:

- The mediating link DOD modification strategy transfers the modification strategy (DOD modification strategy) from input interface of DOD modification determination to the input interface of method determination.
- The mediating link modification focus to method transfers modification foci (DOD modification focus) from the input interface of DOD modification determination to the input interface of method determination.
- The mediating link current DOD basis evaluation to method transfers assessments of a DOD on the basis of design requirements (current DOD basis evaluation) from the input interface of DOD modification determination to the input interface of method determination.
- The mediating link modification focus to assessment point transfers modification foci (DOD modification focus) from the input interface of DOD modification determination to the input interface of assessment point determination.
- The mediating link current DOD contents to assessment point transfers the contents of the current DOD (current DOD contents) from the input interface of DOD modification determination to the input interface of assessment point determination.
- The mediating link current DOD basis evaluation to assessment point transfers assessments of a DOD on the basis of design requirements (DOD assessment) from the input interface of DOD modification determination to the input interface of assessment point determination.
- The mediating link current design requirements to assessment point transfers design requirements (current design requirements) from the input interface of DOD modification determination to the input interface of assessment point determination.
- The mediating link rejected modifications to assessment point transfers rejected foci for modification (rejected DOD modification focus) from the input interface of DOD modification determination to the input interface of assessment point determination.
- The mediating link rejected modifications to extension transfers rejected foci for modification (rejected DOD modification focus) from the input interface of DOD modification determination to the input interface of default extension method.

- The mediating link modification focus to extension transfers modification foci (DOD modification focus) from the input interface of DOD modification determination to the input interface of default extension method.
- The mediating link current DOD contents to extension transfers the contents of the current DOD (current DOD contents) from the input interface of DOD modification determination to the input interface of default extension method.
- The mediating link current DOD contents to result preparation transfers the contents of the current DOD (current DOD contents) from the input interface of DOD modification determination to the input interface of modification determination result preparation.
- The private link modification method control to assessment point transfers control information for modification methods (modification method) from the output interface of method determination to the input interface of assessment point determination.
- The private link modification method control to extension transfers control information for modification methods (modification method) from the output interface of method determination to the input interface of default extension method.
- The private link modification method evaluation transfers expected modification impacts (expected modification impact) from the output interface of assessment point determination to the input interface of method determination.
- The private link to be realised assessment points transfers assessment points which need to be realised (assessment points to be realised) from the output interface of assessment point determination to the input interface of default extension method.
- The private link extension results transfers results from the extension method (addition results) from the output interface of default extension method to the input interface of modification determination result preparation.
- The mediating link modification status transfers status information on modification methods (modification status) from the output interface of method determination to the output interface of DOD modification determination.
- The mediating link selected DOD modification transfers selected modifications of a DOD (selected DOD modification) from the output interface of modification determination result preparation to the output interface of DOD modification determination.

The task control within the component DOD modification determination is as follows. Upon activation of DOD modification determination the information links DOD modification strategy, modification focus to method, and current DOD basis evaluation to method are made up-to-date and method determination is activated with task control focus determine modification approach. Upon termination of method determination the information links modification method control to assessment point, modification focus to assessment point, current DOD contents to assessment point, current DOD basis evaluation to assessment point, and current design requirements to assessment point, and rejected modifications to assessment point are made up-to-date and assessment point determination is activated with task control focus determine expected modification impact. Upon termination of assessment point determination, the information link modification method evaluation is made up-to-date and method determination is activated with task control focus determine modification method.

Upon termination of method determination (with task control focus determine modification method) the information link modification method control to assessment point is made up-to-date and assessment point determination is activated with task control focus determine assessment points to realise. Upon termination of assessment point determination (with task control focus determine assessment points to realise) the information links modification method control to extension, to be realised assessment points, modification focus and limitation to extension, rejected modifications to extension, and current DOD contents to extension are made up-to-date and default extension method is activated. Upon termination of default extension method, modification method result preparation is activated and the information links addition results, and current contents to result preparation are made up-to-date.

Upon termination of modification determination result preparation the information links modification status and selected DOD modification are made up-to-date and DOD modification determination terminates itself.

### 9.3.6 Knowledge composition for DOD modification

A number of information types and knowledge bases related to sub-processes of DOD modification are described below. The information type current DOD basis evaluation is depicted in Figure 9.6. This information type contains relations which describe assessments of design requirements on the basis of the current DOD.

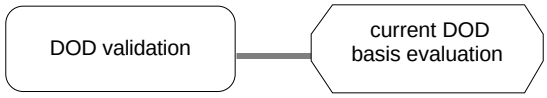


**Figure 9.6** Information type current DOD basis evaluation, among other related to the component DOD validation.

The relations defined in the information type current DOD basis evaluation include:

```
relations
    is_requirement_name_in_use:              requirement_name;
    is_qualified_requirement_name_in_use:    qualified_requirement_name;
    violated_requirement,
    satisfied_requirement:                   requirement_name;
    supported_qualified_requirement,
    undermined_qualified_requirement,
    has_refinement,
    is_a_refinement:                         qualified_requirement_name;
```

The relations is requirement name in use and is qualified requirement name in use represent the names of the current design requirements. The relations violated requirement and satisfied requirement denote whether a requirement is violated or satisfied by the current DOD. The relations supported qualified requirement and undermined qualified requirement denote whether a qualified requirements is supported or undermined by the current DOD. The relations has refinement and is a refinement indicate whether a qualified requirement has refinements or is a refinement of a qualified requirement.

The information type DOD modification focus is depicted in Figure 9.7. This information types refers to two information types: design requirement focus and design object blueprint focus. The information type design requirement focus refers to three information types: possible design requirement focus, selected design requirement focus, rejected design requirement focus. The information type design object blueprint focus refers to three information types: possible design object blueprint focus, selected design object blueprint focus, and rejected design object blueprint focus. Related to these information types is the information type intermediate design object blueprint focus, which contains information on where to put the design object blueprint focus.

The relations defined in the information types related to DOD modification focus are:

```
relations
    qr_possible_as_focus,
    qr_selected_as_focus,
    qr_rejected_as_focus:                    qualified_requirement_name;
    req_possible_as_focus,
    req_selected_as_focus,
    req_rejected_as_focus:                   requirement_name;
    blueprint_possible_as_focus,
    blueprint_selected_as_focus,
    blueprint_rejected_as_focus:             name;
```

The relations qr possible as focus, qr selected as focus, and qr rejected as focus are used to focus on a qualified requirement. The relations req possible as focus, req selected as focus, and req rejected as focus are used to focus on a requirement. The relations blueprint possible as focus,

blueprint selected as focus, and blueprint rejected as focus are used to focus on part of the basic design object information.

**relations**
too_detailed_blueprint_focus: name;
precise_blueprint_focus: name;

The relations too detailed blueprint focus and precise blueprint focus provide information useful for focussing the design object blueprint focus.

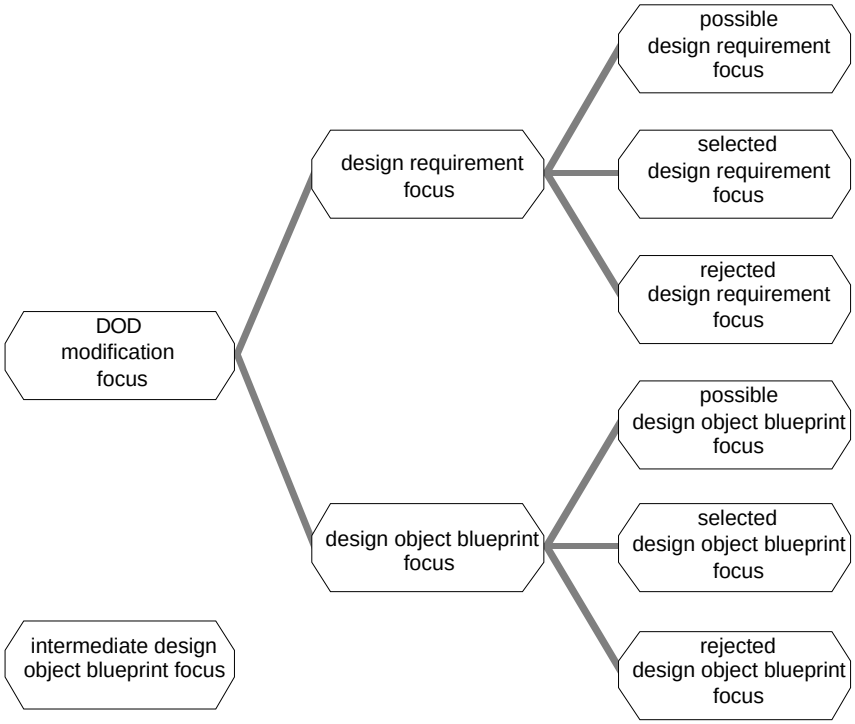


**Figure 9.7** Information type DOD modification focus.

The information type DOD modification strategy is depicted in Figure 9.8. This information type consists of four information types: DOD modification strategy specification, rejected design requirement focus, rejected design object blueprint focus, and rejected DOD modification. The modification strategy is defined in DOD modification strategy specification, and rejected modification foci (both design requirement and design object blueprint) and rejected modifications to a DOD are described in Appendix B.

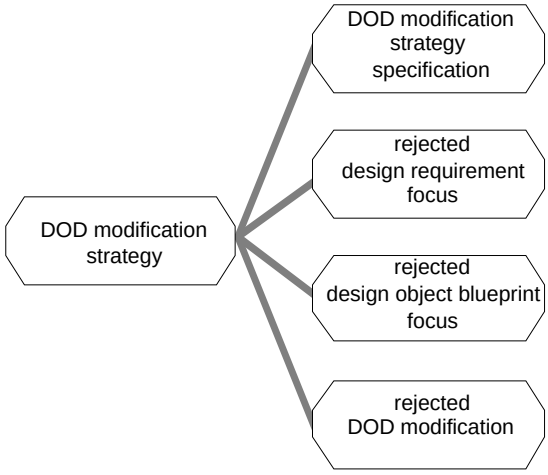


**Figure 9.8** Information type DOD modification strategy.

The relation defined in the information type DOD modification strategy specification is:

**relations**
modification_strategy: modification_method_identification *
modification_method_characterisation;

A modification strategy consists of a modification method identification and a modification method characterisation. Extension of a DOD is an example of modification method identification. Modification method characterisation includes for example an indication of what needs to be focussed on (e.g., undermined qualified requirements, or unresolved (i.e., neither undermined nor supported) qualified requirements).

Information types related to the component DOD modification determination are shown in Figure 9.9. The information type modification status contains information on the results of applying a modification method. The information type modification method contains information on the modification method to apply. The information type modification method evaluation contains an analysis of potential results of a modification method. The information type to be realised assessment points contains information on assessment points which need to become realised (to ultimately realise the satisfaction of a design requirement in focus). The information type extension results contains results of the extension method.

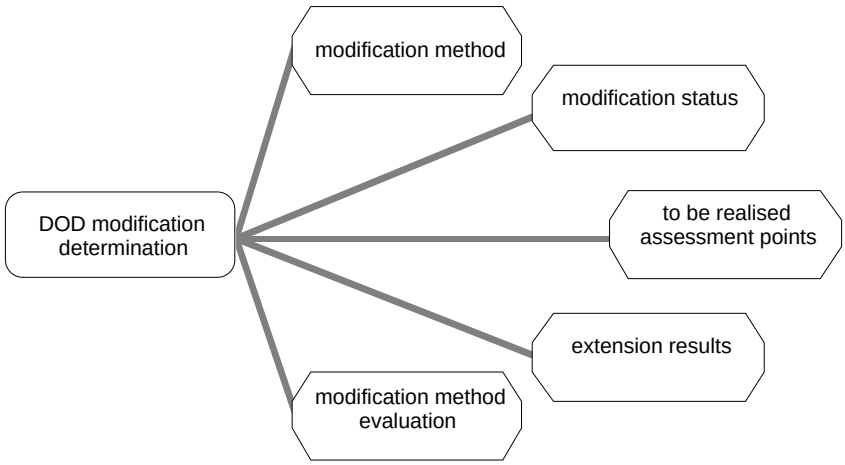


**Figure 9.9** Information types related to DOD modification determination.

Relations defined in the information type modification status are:

**relations**
modification_finished;
modification_consequences_needed;

These two relations express whether a modification step has been finished, or that consequences of the partial modification are needed in order to continue this particular modification step.

The relation defined in the information type modification method is:

**relations**
selected_DOD_modification_method: DOD_modification_method;

This relation indicates which particular modification method is to be applied to the current DOD.

The relations defined in the information type modification method evaluation are:

**relations**
has_expected_modification_impact: design_object_property_atom *
modification_impact ;

The sort design object property atom contains the meta-description of the information type design object information. The relation has expected modification impact denotes what the effect is of realising a design object property (i.e., an assessment point). A modification impact specifies that, e.g., a component is to be defined, an information link is to be created, the mapping of information types within an information link is to be modified.

The relation defined in the information type to be realised assessment points is:

```
relations
    to_be_realised:                 design_object_property_atom;
```

The relation to be realised denotes which design object property (i.e., assessment point) is to be effectuated in the current DOD.

The relations defined in the information type extension results are:

```
relations
    extension:                      design_object_blueprint_atom;
```

The relation extension specifies which information is to be added to the current DOD. The sort design object blueprint atom contains the meta-description of the information type basic design object blueprint information.

A number of knowledge bases related to DOD modification are shown in Figure 9.10. They are described in more detail in the remainder of this section.

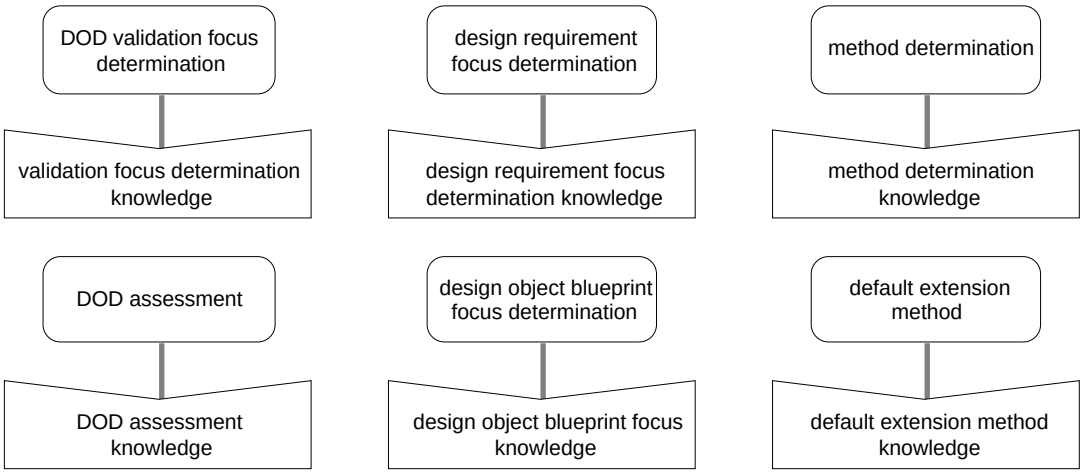


**Figure 9.10** Knowledge bases related to sub-components of DOD modification.

The knowledge base validation focus determination knowledge is employed to relate the contents of a DOD to foci for deductive refinement on the basis of the current design requirements. Instances of knowledge are shown below.

**Example from knowledge base validation focus determination knowledge**

The knowledge element shown specifies that the property to which a requirement refers, related to a qualified requirement, is part of the focus for deductive refinement of the current DOD.

```
if   is_qualified_requirement(   QR: qualified_requirement_name,
                                 Q: qualification,
                                 R: requirement_name )
     and   is_requirement(       R: requirement_name,
                                 A: design_object_derivable_atom )
then      is_part_of_DOD_refinement_focus( A: design_object_derivable_atom, pos );
```

The knowledge base DOD assessment knowledge is employed to assess the current design requirements on the basis of the current DOD. Example knowledge is shown below.

**Example from knowledge base DOD assessment knowledge**

The knowledge below specifies that if the design object property to which a requirement refers is explicitly false in the (deductively refined) current DOD, then this requirement is violated (and therefore not satisfied).

```
if   is_requirement( R: requirement_name, A: design_object_property_atom )
     and   holds( A: design_object_property_atom, neg )
then      violated_requirement( R: requirement_name )
     and   not satisfied_requirement( R: requirement_name );
```

The knowledge base design requirement focus determination knowledge is employed to identify possible foci for modification. An example knowledge element is shown below.

**Example from knowledge base design requirement focus determination knowledge**

The knowledge element below specifies that if the modification strategy involves undermined qualified requirements, and a qualified requirement can be found which is undermined, and this qualified requirement does not have refinements (i.e., it is at the 'bottom' of a qualified requirement refinement tree), then this qualified requirement is a possible focus.

```
if   modification_strategy( I: modification_method_identification, undermined_qr )
     and   undermined_qualified_requirement( QR: qualified_requirement_name )
     and   not has_refinement( QR: qualified_requirement_name )
then      qr_possible_as_focus( QR: qualified_requirement_name );
```

The knowledge base design object blueprint focus determination knowledge is employed to identify possible foci for modification. Example knowledge is shown below.

**Example from knowledge base design object blueprint focus determination knowledge**

The knowledge element below specifies that if a too detailed focus on part of the design object has been given as an intermediate design object blueprint focus, and it can be concluded (by another part of the knowledge base) which components encompass this part of the design object, then this component is a possible focus on part of the design object.

```
if   too_detailed_blueprint_focus( N: name )
     and   has_encompassing_component( N: name, C: component_name )
then      blueprint_possible_as_focus( C: component_name );
```

The knowledge base modification method knowledge is employed to select a specific modification method (a sub-component of DOD modification determination) to modify the current DOD. The component that corresponds to the selected modification method is activated by means of evaluation criteria and task control in the component DOD modification determination. Example instances of knowledge are shown below.

**Parts from knowledge base modification method knowledge**

The first part specifies that if there is more than one modification impact of non-realised assessment points, then this modification step is not finished yet: consequences of the current modification are needed before continuing the realisation of non-realised assessment points.

```
if   more_than_one_non_realized_impact
then      modification_consequences_needed
     and   not modification_finished;
```

The second part specifies that if the modification strategy states that the modification process is to continue modification by means of the already chosen method, then assessment points need to be derived.

```
if   modification_strategy( continue_already_chosen_method, no_criterion )
then      selected_DOD_modification_subtask( derive_assessment_points );
```

The knowledge base extension method knowledge is employed to specify which basic design object information has to be removed from the current DOD. Example parts from this knowledge base are described below.

**Parts from knowledge base extension method knowledge**

The parts below specify the relation between a non-realised assessment point (i.e., a property of the design object) related to basic design object information. The first part specifies that if it needs to be realised that a component C exists, then the fact is_component( C: component_name ) needs to be added to the current DOD.

```
if   to_be_realised( is_component( C: component_name ) )
then      extension( is_component( C: component_name ) );
```

The second part specifies that if a component is to be an agent, then a particular characterisation has to be added to the current DOD.

```
if   to_be_realised( is_an_agent( C: component_name ) )
then      extension( has_characterisation( C: component_name, agent ) );
```

The third part specifies that if an information link is to be created which exists within a component C and connects component D1 to component D2, then three additions need to be made to the current DOD.

```
if   to_be_realised( link_connects(                  I: information_link_name,
                                                     C: component_name,
                                                     D1: component_name,
                                                     D2: component_name ) )
then      extension( has_information_link(           C: component_name,
                                                     I: information_link_name ) )
     and  extension( has_source_component(           I: information_link_name,
                                                     D1: component_name ) )
     and  extension( has_destination_component(      I: information_link_name,
                                                     D2: component_name ) );
```

## 9.4 Refinement of DODM History Maintenance

Both the process composition and the knowledge composition for the process RQSM history maintenance are described below.

### 9.4.1 Process composition for DODM history maintenance: identification of processes and abstraction levels

The refinement of the process of DODM history maintenance presented in this section is similar to the refinement of the process of RQSM history maintenance: similar processes and input & output information types are identified (with 'DOD' in the name instead of 'RQS') with similar content (i.e. in this case descriptions of design objects).

The process of DODM History Maintenance is responsible for maintaining and retrieving information of the DODM process for future use. In particular information on the design object descriptions considered (accepted and/or rejected) and information on states within the DOD modification process are stored. A number of sub-processes are performed to this purpose, as shown in Figure 9.11. The process DODM history maintenance is composed of the process DOD history maintenance, DOD assessment & RQS history maintenance, and DOD modification state history maintenance.

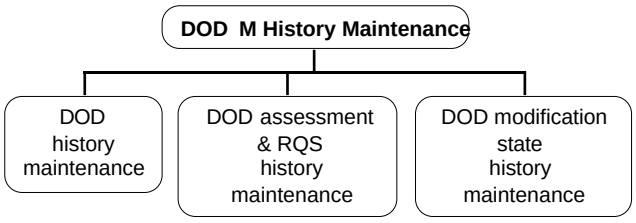


**Figure 9.11** Process composition for DODM History Maintenance.

The process DOD history maintenance is responsible for storing, retrieving and managing descriptions of design objects over time. The process DOD assessment & RQS history maintenance is responsible for storing, retrieving, and managing DOD assessments and sets of qualified requirements. The process DOD modification state history maintenance is responsible for storing, retrieving and managing modification states (i.e., information regarding the modification processes). This process is similar to the process DOD history maintenance. These three processes are further described in Section B.2.3.

The *interface information types* of the processes distinguished within the process DODM History Maintenance are listed in Table 9.5 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| DOD history maintenance | • DOD<br>• current DOD contents<br>• DOD history queries<br>• current DOD replacement request | • DOD history search results<br>• current DOD replacement results<br>• new current DOD contents<br>• current DOD name |
| DOD assessment & RQS history maintenance | • RQS<br>• RQS history queries<br>• DOD assessment<br>• DOD assessment history queries | • RQS history search results<br>• DOD assessment history search results |
| DOD modification state history maintenance | • given current DOD name<br>• overall design strategy<br>• DOD modification state history queries<br>• DOD modification progress | • DOD modification state history search results |

**Table 9.5** Interface information types for sub-processes of DODM History Maintenance.

- The process DOD history maintenance needs information on (given) design object descriptions (DOD), the contents of the current DOD (current DOD contents), queries on DOD history (DOD history queries), and requests for replacement of the current DOD (current DOD replacement request). This process generates results of searching the DOD history (DOD history search results), results on the success of replacing the current DOD (current DOD replacement results), new contents for the current DOD (new current DOD contents), and the name of the newest DOD (current DOD name).
- The process DOD assessment & RQS history maintenance requires sets of qualified requirements (RQS), queries on RQS history (RQS history queries), assessments of design object descriptions (DOD assessment), and queries on DOD assessment (DOD assessment history queries). This process generates results of searching the RQS history (RQS history search results), and results of searching the DOD assessment history (DOD assessment history search results).
- The process DOD modification state history maintenance requires information on the name of the newest DOD (given current DOD name), overall design strategy (overall design strategy), queries on DOD modification state information history (DOD modification state history queries), and progress of the modification process (DOD modification progress). The process DOD modification state history maintenance produces results of searching the DOD modification state history (DOD modification state history search results).

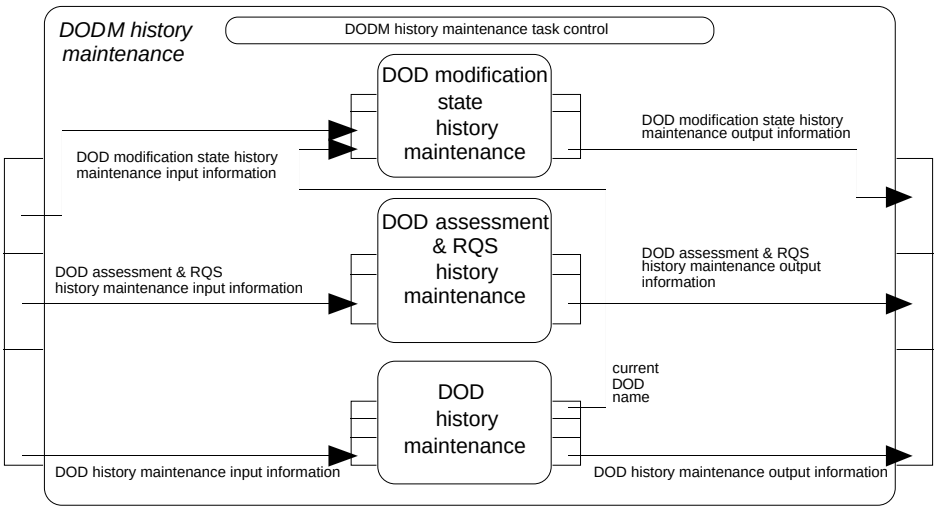


**Figure 9.12** Information links within the process of DODM history maintenance.

### 9.4.2 Process composition relation within DODM history maintenance

The information links in the component DODM history maintenance are shown in Figure 9.12. Within this component six mediating links and one private link are defined:

- The mediating link DOD modification state history maintenance input information transfers information expressed in overall design strategy, DOD modification state history queries, and DOD modification progress from the input interface of DODM history maintenance to the input interface of DOD modification state history maintenance.
- The mediating link DOD assessment & RQS history maintenance input information transfers information expressed in RQS, RQS history queries, DOD assessment, DOD assessment history queries from the input interface of DODM history maintenance to the input interface of DOD assessment & RQS history maintenance.
- The mediating link DOD history maintenance input information transfers information expressed in DOD, current DOD contents, DOD history queries, and current DOD replacement request from the input interface of DODM history maintenance to the input interface of DOD history maintenance.
- The private link current DOD name transfers the name of the DOD currently being stored (current DOD name) from the output interface of DOD history maintenance to the given name of the DOD currently being stored (given current DOD name) at the input interface of DOD modification state history maintenance on the basis of an explicit mapping between these information types.
- The mediating link DOD modification state history maintenance output information transfers information expressed in DOD modification state history search results from the output interface of DOD modification state history maintenance to the output interface of DODM history maintenance.
- The mediating link DOD assessment & RQS history maintenance output information transfers information expressed in RQS history search results, and DOD assessment history search results from the output interface of DOD assessment & RQS history maintenance to the output interface of DODM history maintenance.
- The mediating link DOD history maintenance output information transfers information expressed in DOD history query results, current DOD replacement results, and new current DOD contents from the output interface of DOD history maintenance to the output interface of DODM history maintenance.

The task control within the component DODM history maintenance is as follows. DODM history maintenance can be activated with one of the following task control foci: initial storage of information, continued storage of information, update of the history, execute queries, or replacement of current DOD preparation. For four of these task control foci the same components and information links are activated, the task control focus replacement of current DOD preparation results in a slightly different activation of components and information links:

- On activation with one of the task control foci initial storage of information, continued storage of information, update of the history, and execute queries the information links DOD modification state history maintenance input information, DOD assessment & RQS history maintenance input information, and DOD history maintenance input information are made up-to-date, and DOD history maintenance is activated with the same task control focus as DODM history maintenance. Upon termination of DOD history maintenance, DOD assessment & RQS history maintenance is activated with the same task control focus as DODM history maintenance. Upon termination of DOD assessment & RQS history maintenance, DOD modification state history maintenance is activated with the same task control focus as DODM history maintenance and the information link current DOD name is made up-to-date. Upon termination of DOD modification state history maintenance, the information links DOD modification state history maintenance output information, DOD assessment & RQS history maintenance output information, and DOD history maintenance output information are made up-to-date and DODM history maintenance terminates itself.

- On activation with the task control focus replacement of current DOD preparation, the information link current DOD history maintenance input information is made update, and DOD history maintenance is activated with task control focus replacement of current DOD preparation. Upon termination of DOD history maintenance the information link DOD history maintenance output information is made up-to-date and DODM history maintenance terminates itself.

### 9.4.3 Knowledge composition for DODM history maintenance

Most of the information types and knowledge bases related to the process DODM history maintenance are not refined, one exception is the information type DOD modification state information, as shown in Figure 9.13.

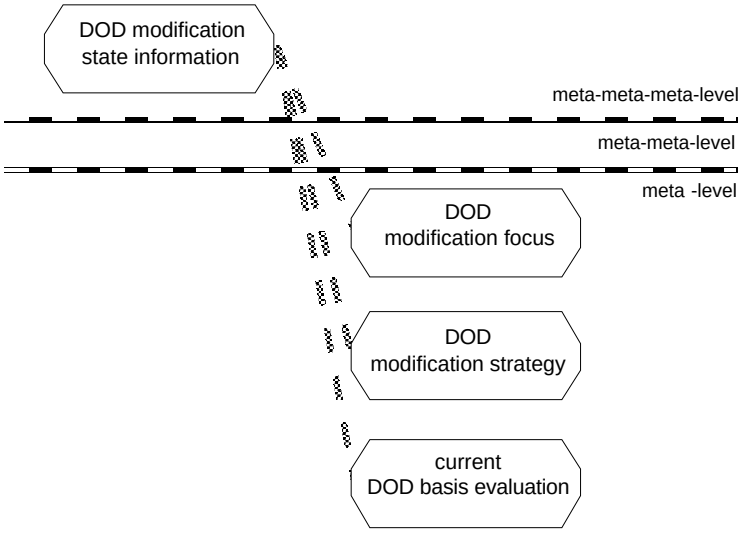


**Figure 9.13** Refinement of the information type DOD modification state information.

The meta-descriptions of the information types DOD modification focus, DOD modification strategy, and current DOD basis evaluation are related to the sort DOD modification state attribute value (see Section A.2.3), thereby instantiating the relation has DOD modification state value.

## 9.5 Discussion

In this chapter the results of the previous chapters are extended: the generic model for design is further refined resulting in the model for re-design of compositional systems. The refinement of the process DOD manipulation is addressed in this chapter. Four refinement steps have been described, as depicted in Figure 9.14.

In the first refinement step the information types and knowledge-base related to the process current DOD maintenance are instantiated. In the second refinement step the information types and knowledge base related to the process deductive DOD refinement are instantiated. In the third refinement step the process DOD modification is partially refined; additional refinements of sub-processes of DOD modification are described in Appendix B. In the fourth refinement step the process DODM history maintenance is partially refined; additional refinements of sub-processes of DODM history maintenance are described in Appendix B.

The description of the refinement of DOD manipulation is the realisation of the desideratum dr3 (“model of the manipulation of compositional system structures”).

The refinement of the process DOD manipulation described in this chapter is more detailed than the initial description of the structure of DOD modification (which was also the basis for RQS modification), described in (Brazier, Langen, Treur and Wijngaards, 1996); the refinement of a modification process is depicted in Figure 8.15. In (Brazier, Langen, Treur and

Wijngaards, 1996) four processes are distinguished as refinements of the process DOD modification:

- analysis of current description, that investigates which conflicts, unsatisfied design requirements, et cetera, are present in the current DOD,
- modification focus determination, that determines which parts of the current DOD must be modified to be able to resolve the identified problems,
- modification method determination, that determines the method for modifying the parts of the current DOD that are in focus,
- modification according to method, that modifies the parts of the current DOD that are in focus, according to the method determined.

The processes distinguished above can be found in the compositions of DOD modification (Section 9.3). The analysis of current description is located in DOD validation; modification focus determination is located within DOD modification focus identification; modification method determination is located within DOD modification determination; and modification according to method is realised by task control knowledge and several processes corresponding to methods in DOD modification determination.

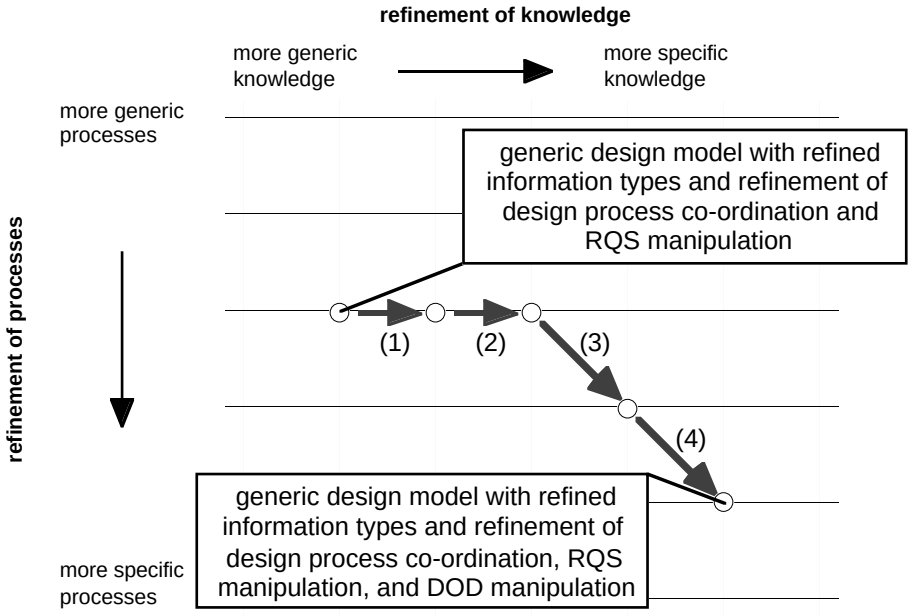


**Figure 9.14** Refinement steps described in this chapter: the numbered arrows correspond to sections 9.1 to 9.4.

The process composition and knowledge composition described in this chapter are generic in the sense that more elaborate, knowledge-intensive specialisations can be added. For example, in the process specialisation for DOD modification determination additional modification methods can be added.

Part **IV**

# Examples of Re-design of Compositional Systems

In this part examples are given of the application of the model for re-design of compositional systems. In Chapter 10 a model for a design agent is described, based on the model for re-design of compositional systems and the agent model (described in Chapter 4). In Chapter 11 the re-design model is applied to the domain of diagnostic reasoning systems, resulting in the re-design of a specific diagnostic reasoning system. In Chapter 12 the design agent is applied to the domain of self-modifying multi-agent systems, resulting in the re-design of the multi-agent system by one of its agents.

# 10 An Agent Model for Dynamic Re-design

Within multi-agent systems, design is a task often performed by one or more specialised agents. Other agents interact with such 'design agents' by, for example, providing qualified requirements, initial (partial) design object descriptions, and design process objectives. Specialised agents are often encountered in human society: e.g., architects are specialised agents: their area of expertise is the design of buildings. A design agent designs on the basis of the information received from other agents, and makes results of the design process available to other agents.

A generic agent model is described in Section 4.5.2. A generic model for design is described in Chapter 6, and a refinement of this model in Chapters 7, 8, and 9. The resulting models are used in this chapter to construct a generic model for a design agent. This generic design agent is applied to the re-design of a multi-agent system. That is, the generic design agent is refined for the domain of re-design of compositional systems.

Two approaches can be used to construct a design agent for the re-design of compositional systems. The first approach is to combine the re-design model of compositional systems with the generic agent model, resulting in a model for a design agent for re-design of compositional systems. The second approach is to combine the generic design model with the generic agent model, resulting in a generic model for a design agent, which is then refined to a model for a design agent for re-design of compositional systems by applying the refinement of the design model as described in Chapters 7, 8, and 9. The second approach, in which an intermediate generic design agent is constructed, is taken in this chapter.

The process of constructing a design agent is a design process on its own: design process objectives are distinguished, as are qualified requirements. This chapter is organised according to the three main processes within the process of design. In Section 10.1 design process objectives are described for the design of a design agent. In Section 10.2 requirements for the design of a design agent are formulated. These requirements guide the construction process. In Section 10.3, the generic model for design (see Chapter 6) is combined with the generic agent model (see Section 4.5.2), resulting in a generic model for a design agent. In Section 10.4, the generic model for a design agent, described in Section 10.3, is specialised for re-design of compositional systems (using the model described in Chapters 7, 8, and 9). In Section 10.5 the models described in this chapter are discussed.

## 10.1 Design process objectives on a design agent

Objectives may be defined for the design process with which a design agent is designed. These design process objectives may influence choices with respect to specific design strategies.

The following design process objectives are distinguished:

- *A generic agent model is to be the basis of the design of the agent architecture.* The generic agent model from (Brazier, Jonker and Treur, 1996) and described in Section 4.5.2, is used to model the 'agent' process of the design agent.
- *A generic design model is to be the basis for the design of the design task of the design agent*. The generic model for design, described in Chapter 6, is used to model the 'design' process performed by the design agent.
- *Minimal modifications to the agent model.* The agent model should be modified as little as possible.
- *Minimal modifications to the design model.* The design model should be modified as little as possible.

- *Minimal modifications to the agent model are preferred over minimal changes to the design model*. If modifications are necessary, these should preferably be made to the design model instead of the agent model.
- *An intermediate, generic design-agent is designed before a specialised design agent.* Two agent models have to be produced: first a model of a generic design agent, and then a model of a specialised design agent.

## 10.2 Requirements on a design agent

A generic model for an agent usually supports communication and/or world interaction about topics related to its specific tasks. A generic model for a *design agent*, needs to integrate the design process and the internal processes of an agent, providing support for communication and world interaction on issues related to a design process.

In this thesis a generic agent model and a generic design model are combined to form a new, generic model, for a design agent. A distinction can be made between design agents in general (see Section 10.3), and design agents specialised in a specific task, i.e., re-design of compositional structures (see section 10.4). For both types of design agents requirements can be formulated which need to be fulfilled by models of these types of design agents.

### 10.2.1 Requirements on a generic model for a design agent

Requirements on a generic model for a design agent can be split into two categories: requirements on the 'agent' properties of the design agent, and requirements on the integration of the 'design properties' in the design agent.

A generic design agent should have the agent abilities:

- *is capable of bi-directional communication.* A design agent has to be able to bi-directionally communicate about information needed by, or resulting from, a design activity.
- *is capable of world interaction.* A design agent has to be able to interact in the material world to observe (or provide) information needed by (or resulting from) a design activity.
- *is capable of co-operation.* A design agent has to be able to co-operate (and, e.g., to negotiate) on a design activity.
- *is capable of agent own process control.* A design agent has to be able to monitor and plan its own processes, including the design process at a global level.

The agent own process control does *not* cover control of the processes *inside* the design process except by providing design process objectives.

The desired properties on the design model are:

- Explicit distinction between the manipulation of design object descriptions, manipulation of sets of qualified requirements, and co-ordination of the design process.
- Explicit representation and manipulation of design process objectives.
- Explicit representation and manipulation of (sets of) qualified requirements.
- Explicit representation and manipulation of design object descriptions.

Requirements on a generic model for a design agent also address the integration of a process of design within an agent model. The following requirements can be formulated:

- *The design process is to be modelled within the agent as one of its (possible) capabilities.* The ability to perform a design process is to be modelled as (one of) the agent's specific tasks.
- *Information needed for the design process can be acquired via communication or world interaction.* Information on which a specific design process is based (design process objectives, sets of qualified requirements, design object descriptions) can be acquired by two means: by communication, or by observation in the material world.
- *Information resulting from the design process can be made available via communication or world interaction.* All types of information resulting from a design process (design object description information, requirement qualification set information, process results,

design process evaluation status) can be made available by two means: by communication, or by actions in the material world.

### 10.2.2 Requirements on a model for a specialised design agent

Requirements on a model for a design agent specialised for re-design of compositional structures address the refinement of the design task: both the refinement of processes and the refinement of knowledge structures. The refinement of the generic design model, described in Chapters 7, 8, and 9, is employed in the design agent. The following requirements can be formulated which supplement the requirements formulated in the previous sub-section:

- *The design process is to be refined to design compositional systems.* The design process needs to focus on the design of compositional systems.
- *A knowledge-intensive system is to be represented as a design object description in the design process.* The design object descriptions are to be descriptions of knowledge-intensive systems.
- *Qualified requirements on compositional systems are to be represented in the design process.* The qualified requirements within the design process are to be qualified requirements on compositional systems.
- *The manipulation of compositional systems descriptions is to be modelled in the design process.* The design process is to include the manipulation of compositional systems, on the basis of qualified requirements on compositional systems.
- *The manipulation of qualified requirements on compositional system is to be modelled in the design process.* The design process is to include the manipulation of qualified requirements.
- *The design process is to include knowledge on the co-ordination of the design process.* The design process is to include knowledge on the co-ordination of the design process specialised for re-redesign of compositional systems.

The purpose of these requirements is to guide the construction of a generic model for a design agent, specialised in re-design of compositional structures. This generic model can be applied in different situations (for example, dynamic design of agents (i.e. self-modifying multi-agent system) and distributed design of a knowledge-intensive system), in which the design agent has different roles and/or abilities.

## 10.3 Constructing a generic model for a design agent

A generic model for a design agent is constructed by combining two existing generic models: the generic design model (Chapter 6) and the generic agent model (Section 4.5.2). The resulting model is described in three phases: first the process composition is described, then the knowledge composition, and finally the relation between process composition and knowledge composition. Only modified parts of the generic agent model are described in this section.

### 10.3.1 Process composition

The process composition of the design agent is described by the identification of processes at different abstraction levels, after which the composition relation of these processes is specified.

**Identification of processes at different abstraction levels.** The design process is placed within the agent specific task. This results in the process composition shown in Figure 10.1.

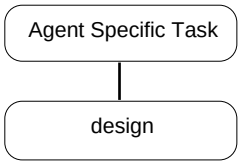


**Figure 10.1** Partial view on the process composition within Agent Specific Task.

The sub-processes within the process of design are not depicted in Figure 10.1, but are the same as the processes described in Section 6.1. Process composition includes descriptions of interfaces of processes. The interface of the design component has three different meta-levels in its interface. The interface of the agent specific task component, however, has one meta-level.

Two modelling options are possible to facilitate information exchange between processes within the agent and the design component within the agent specific task.

- *Modifying the design component.* A translation is made between information needed for, or provided by, the design process, by mapping several meta-levels into one level of information, which can then be used in the interface of the agent specific task.
- *Modifying the agent.* The agent specific task has three meta-levels in its interface, corresponding to the meta-levels in the interface of the design task, and other processes within the agent process and the agent process itself also have three meta-levels in its interfaces.

A (preferred) minimal change to the 'agent' part of the design agent is achieved by adopting the first solution: by using information links to transfer information between meta-levels. Some modifications may, however, be needed to information types in levels distinguished in the interface of the design process.

The information types in the interfaces of Agent Specific Task and Design are described in Table 10.1. Note that the input and output interface of Design have been slightly modified: two information types have been added; the existing information types did not provide the extra meta-level needed to map the information type DOD to a higher level.

In Table 10.1 information types in the interface of the component Agent Specific Task and its sub-component Design are described.

| *process* | *input information type* | *output information type* |
|---|---|---|
| Agent Specific Task | • design input information | • design output information |
| *Within Agent Specific Task* | | |
| Design | • design process objectives<br>• RQS<br>• initial DOD*<br>• DOD | • design process evaluation<br>• RQS assessment<br>• DOD assessment<br>• RQS<br>• resulting DOD*<br>• DOD |

**Table 10.1** Interface information types of component Agent Specific Task and its sub-component Design. New information types are denoted with a *****.

The interfaces of the components described in Table 10.1 are discussed below.

The Agent Specific Task requires information needed for a design process (design input information) and provides results of that design process (design output information).

The task Design needs information on:

- design process objectives (design process objectives),
- sets of qualified requirements (RQS) and a meta-level description of design object descriptions (initial DOD), and
- design object descriptions (DOD).

The task Design produces information on:

- information on evaluations of the design process (design process evaluation),
- assessments of sets of qualified requirements (RQS assessment) and assessments of design object descriptions (DOD assessment), resulting sets of qualified requirements (RQS) and a meta-level description of resulting design object descriptions (resulting DOD), and
- resulting design object descriptions (DOD).

Note that the two (new) information types in the interface of the component Design are meta-descriptions of other information types; enabling transfer of information from three different meta-levels to information at one meta-level.

**Composition relation.** In this section the composed processes identified above are described from a static point of view, i.e., information exchange in a composed process, and from a dynamic point of view, i.e., task control of a composed process.

*Static perspective on process composition.* The information exchange within the process Agent Specific Task is shown in Figure 10.2.

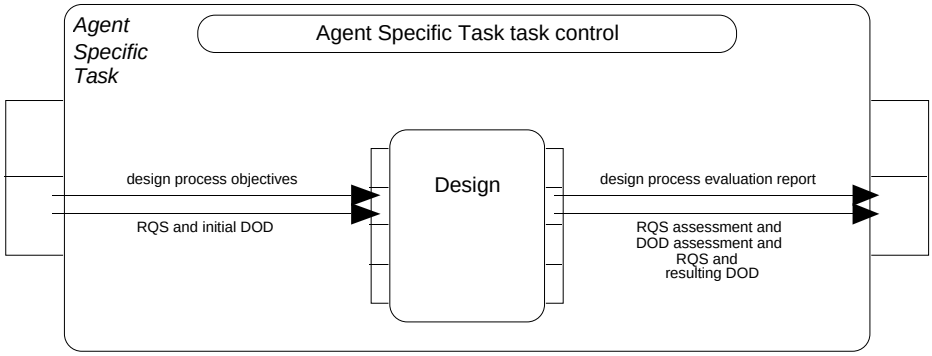


**Figure 10.2** Information exchange within the component Agent Specific Task.

Within the component Agent Specific Task (Figure 10.2) four mediating links are defined:

- The mediating link design process objectives transfers design process objectives from the input interface of Agent Specific Task to the input interface of Design.
- The mediating link RQS and initial DOD transfers meta-information on RQS and initial DOD from the input interface of Agent Specific Task to the standard meta-level description of information of the second meta-level in the input interface of Design on the basis of an explicit mapping between these information types.
- The mediating link design process evaluation transfers design process evaluation from the output interface of Design to the output interface of Agent Specific Task.
- The mediating link RQS assessment and DOD assessment and RQS and resulting DOD transfers from the standard meta-level description of information at the second meta-level from the output interface of Design to resulting level 2 design output at the output interface of Agent Specific Task on the basis of an explicit mapping between these information types.

Figure 10.3 depicts the information links in the component Design and the modified information links. The modified information links are described below.

- The mediating link initial DOD transfers initial DOD from the input interface of Design to the standard meta-level description of information of the first meta-level in the input interface of DOD Manipulation on the basis of an explicit mapping between these information types.
- The mediating link DOD transfers the standard meta-level description of information at the first meta-level from the output interface of DOD Manipulation to resulting DOD in the output interface of Design on the basis of an explicit mapping between these information types.

*Dynamic perspective on process composition.* Task control knowledge within the component Agent Specific Task is straightforward: whenever the component Agent Specific Task becomes active, input information is transferred to the component Design via the information links design process objectives, and initial RQS and initial DOD, after which component Design is activated. Upon termination of the component Design, the information links design process evaluation, RQS assessment, DOD assessment, RQS, and DOD are made ‘up-to-date’ and the component Agent Specific Task terminates itself.

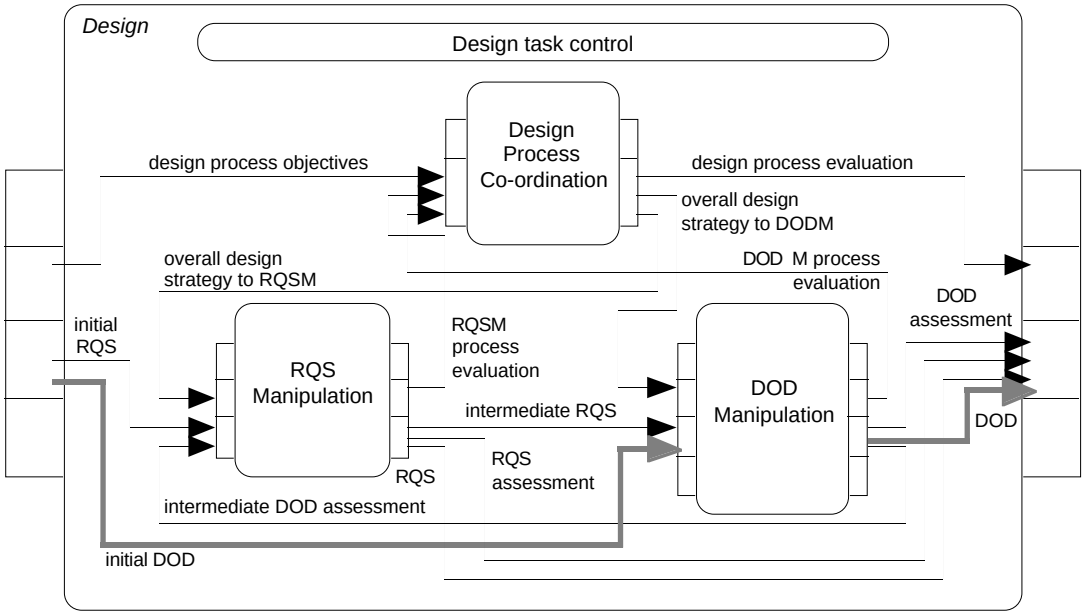


**Figure 10.3** Modified information exchange within the component Design.

### 10.3.2 Knowledge composition

The knowledge composition of the design agent includes information types, knowledge bases, and levels of knowledge abstraction.

**Information types.** The information types in the generic model for the agent and the generic model for design are also in the model for the design agent. In addition, information types are needed to 'connect' the information from the generic model for the agent and the generic model for design.

Table 10.1 depicts the information types in the interface of the sub-component Design of the component Agent Specific Task. The information types Design Input Information and Design Output Information contain information of the design input and output information at three meta-levels. Several smaller information types are needed to construct these two information types.

**Knowledge bases.** The knowledge bases in the generic model for the agent and the generic model for design are also in the model for the design agent.

**Levels of knowledge abstraction.** First compositionality of information types is described, then compositionality of knowledge bases.

In Figure 10.4 a partial view is shown on levels of knowledge abstraction for the information type Design Input Information. Both composition by reference and composition by meta-description are shown in this figure.

The mapping of two meta-levels into one other meta-level (the third meta-level) is shown in Figure 10.4. The information type DOD is mapped into the information type meta level information of DOD (which resides at the next meta-level). The information type initial DOD refers to this information type. Both the information types initial DOD and RQS are mapped into the information type meta level information of RQS and initial DOD (at the third meta-level). The information type RQS and initial DOD information refers to this information type. The information type design input information refers to both this latter information type and the information type design process objectives.

Figure 10.5 shows a partial view of the levels of knowledge abstraction for the information type design output information. Both composition by reference and composition by meta-description are shown in this figure.

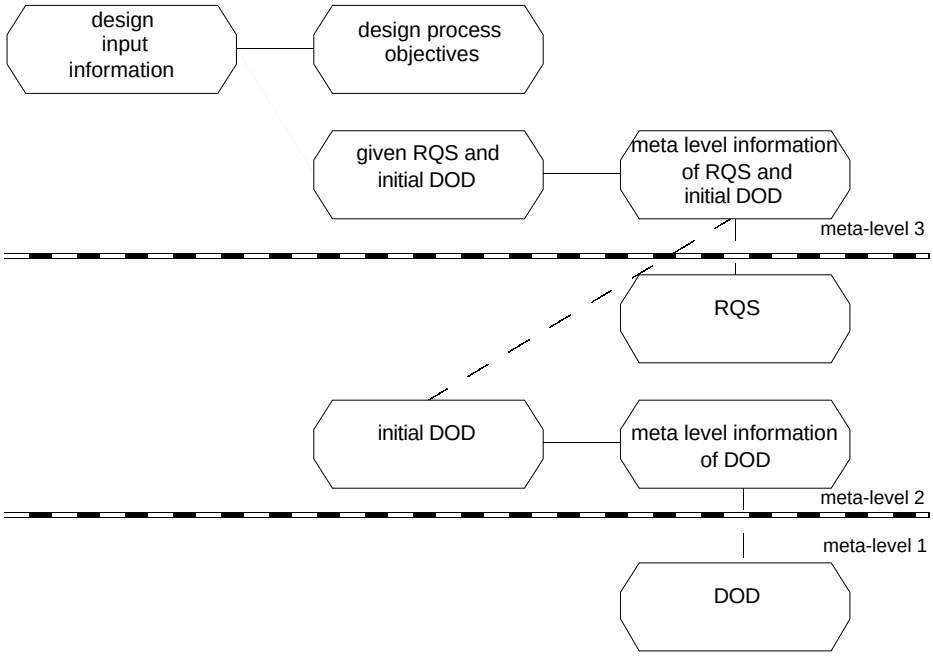


**Figure 10.4** Partial view on levels of knowledge abstraction for information types related to design input information.

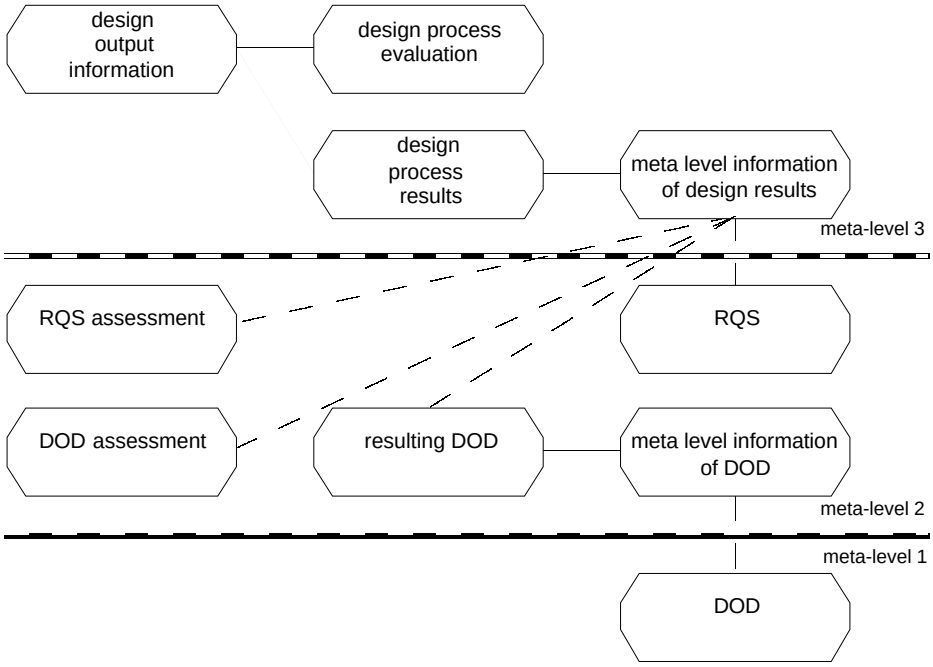


**Figure 10.5** Partial view on levels of knowledge abstraction for information types related to design output information.

The mapping of two meta-levels into one other meta-level (the third meta-level) is shown in Figure 10.5. The information type DOD is mapped into the information type meta level information of DOD (which resides at the next meta-level). The information type resulting DOD refers to this information type. The information types resulting DOD information, RQS assessment, DOD assessment, and RQS are mapped into the information type meta level information of design results (at the third meta-level). The information type design process results refers to this information type. The information type design output information refers to both this latter information type and the information type design process evaluation.

### 10.3.3 Relations between process composition and knowledge composition

The information types related to the Agent Specific Task and Design are depicted in Figure 10.6 and Figure 10.7. The relation between other processes (in both the agent and design) and knowledge structures is not modified.

In Figure 10.6 the process Agent Specific Task is shown in relation to information types. It is a composed process, and therefore does not have a knowledge base of its own.

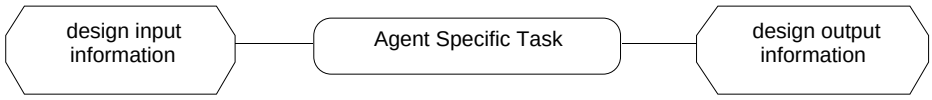


**Figure 10.6** Relation between the process Agent Specific Task and information types in its interface.

In Figure 10.7 the process Design is shown in relation to information types. The extra added information types are indicated. The process Design is a composed process, therefore no knowledge base is indicated.

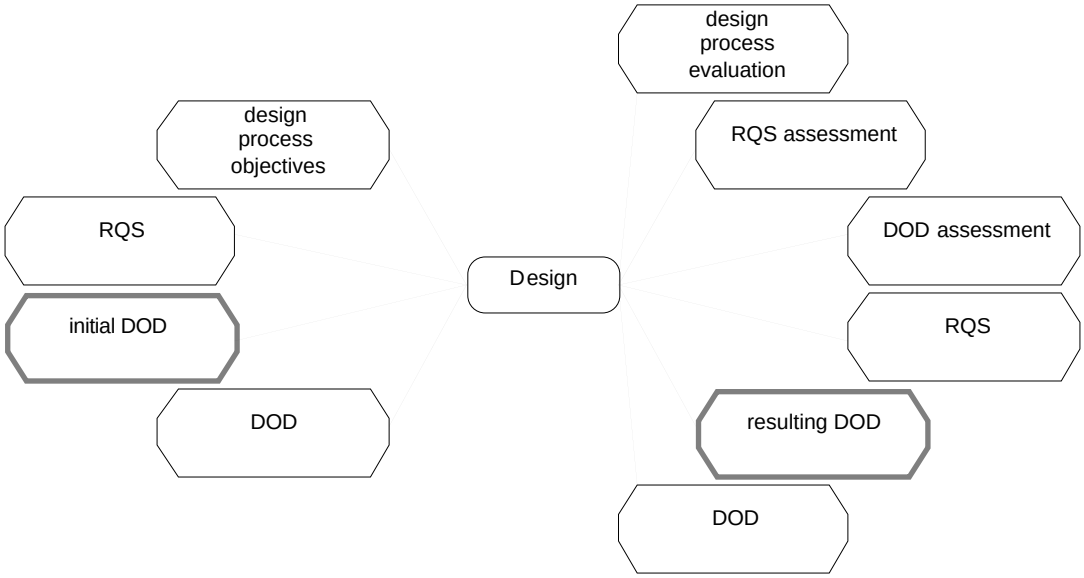


**Figure 10.7** Relation between the process Design and information types in its interface, the new information types are indicated by a thicker outline.

## 10.4 A model for an agent for the re-design of compositional systems

The generic model for a design agent, described in the previous section, has to be refined for the re-design of compositional systems. The design process within the design agent is refined in the same manner as a generic model for design is refined for the re-design of compositional structures; see Chapters 7, 8, and 9 for more details.

The refinement of the knowledge structures, described in Chapters 7, 8, and 9, is also in effect, causing an indirect refinement of all information types which refer (indirectly) to an information type which is specialised. This is a useful feature, e.g., the information which can be communicated to and from agents now includes information on the re-design of compositional structures.

The idea of self-modification is illustrated in Figure 10.8. The box on the left contains the multi-agent system (consisting of three agents and the external world) before modification. The box on the right depicts the multi-agent system after modification (with new agents C and D, and agent B removed). The agent Design Agent causes the modification of the multi-agent system: it reasons about the description of the current multi-agent system, draws a plan to

modify this description, and effectuates the modifications resulting in a modified multi-agent system. The cloud depicts this reasoning process.

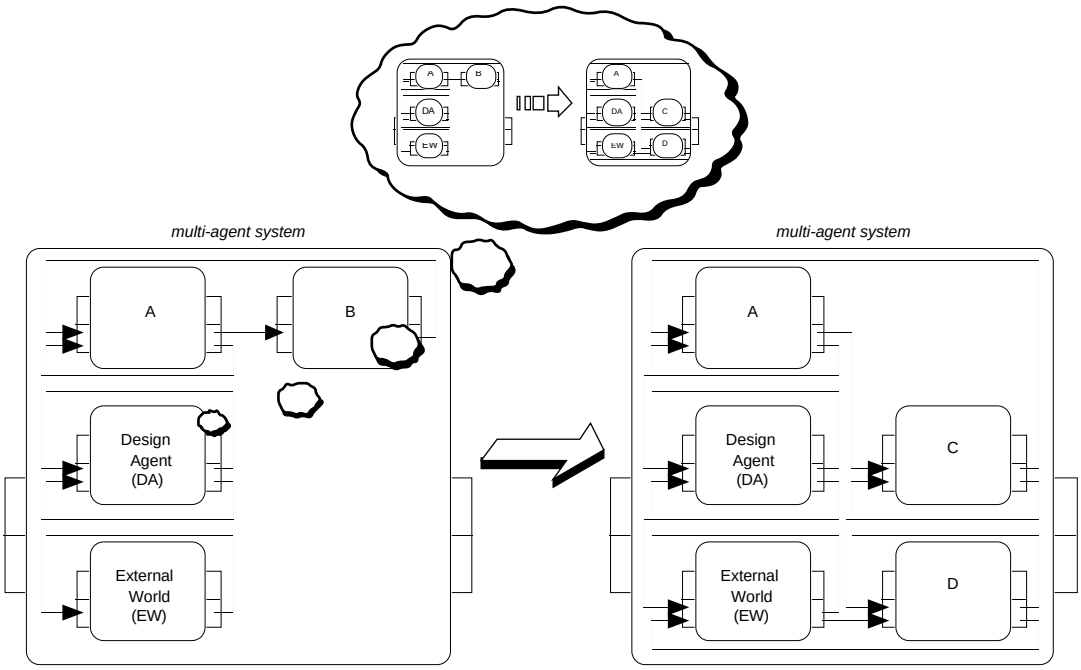


**Figure 10.8** Self modifying multi-agent system: the design agent modifies the structure of the multi-agent system by creating two new agents C and D and removing agent B.

A modelling approach in which a description of a system is manipulated by the system itself (at run-time) is the *mind-matter* approach, introduced in (Jonker and Treur, 1997). This involves two representation relations between a description of a system and the system itself: the description (symbolically) represents the system, and the description is (materially) represented within the system. Modifications to the material representation of the system influence the system (for example, adding a new ability to an agent, makes it possible for that agent to, e.g., reason about new concepts). The design agent plans modifications to the description of a multi-agent system with the possible effect depending on the type of modification: when the changes are effectuated new agents are created, existing agents modified, existing agents removed entirely, and the external world modified.

The combination of a design agent and the mind-matter modelling approach makes it possible to have a conceptually and semantically transparent model of a self-modifying multi-agent system. Tools exist to effectuate this dynamic mind-matter approach by performing material actions in the external world: changes to descriptions of a system entail changes to the system itself.

## 10.5 Discussion

Two generic models, a generic agent model and a generic design model, have been combined to form a generic model for a *design agent*. Based on this model for a design agent, a model for an agent specialised in re-design of compositional systems has been constructed (based on a refinement of the generic model of design).

Requirements have been formulated for a design agent in general, and in addition for a design agent for re-design of compositional structures. These requirements guide the model construction process. The construction process and models developed (a model for a design agent and a model for a design agent for re-design of compositional structures) fulfil all of the formulated design process objectives and requirements.

The model for a design agent is generic with respect to its domain of application, yet is specific with respect to the processes distinguished within the agent (as compared to the generic

agent model). The model for a design agent for re-design of compositional systems is specific with respect to the domain of *design processes*, and specific with respect to the distinguished knowledge structures. These transitions in genericity in two dimensions (process vs. knowledge) are depicted in Figure 10.9.

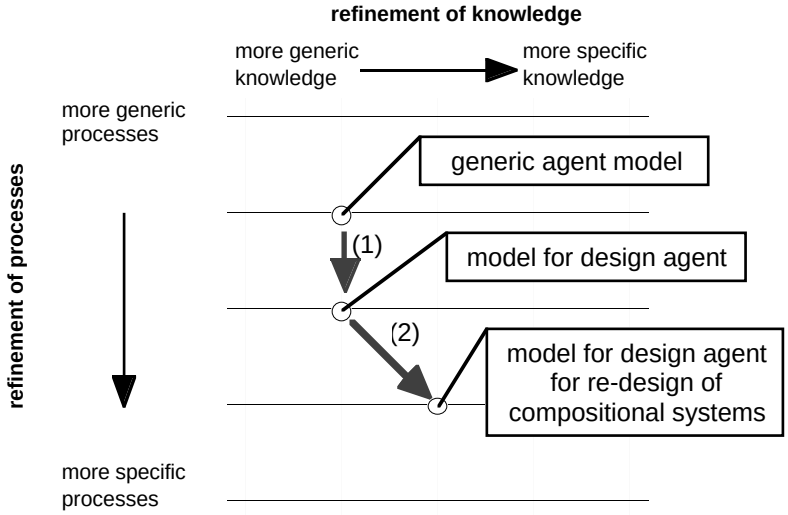


**Figure 10.9** Overview of refinement relations between the generic agent model, the model for the design agent, and the model for the design agent for re-design of compositional systems. The numbered arrows correspond to the two phases during the refinement of the generic agent model.

All three models placed in the matrix in figure 10.9 are *generic* models, yet their 'degree of genericity' differs, as can be inferred from the position of these three models in the matrix.

The construction process for the model of the design agent was straightforward. The position of the design process within the agent was relatively simple: minimal changes to the agent model implies that additional components are placed *within* existing components of the agent. The only serious work was encountered in the mappings between information at one meta-level for agent processes and information at three meta-levels for the process of design.

Adding the specialisation of the design process to the generic model of the design agent was straightforward as well. The specialisation of the process of design, described in Chapters 7, 8, and 9, could be re-applied to the design process in the design agent.

The design agent model described in this chapter is similar, yet more generic and oriented towards multi-agent systems, than the Single Function Agents approach (Dunskus, Grecu, Brown and Berker, 1995; Berker and Brown, 1996) in which specialised design agents, with a particular view on a design object, negotiate with each other to achieve a design object which satisfies requirements imposed by all design agents.

An application of the design agent described in this chapter is discussed in Chapter 12: re-design of a multi-agent system. In the next chapter an application of the re-design model is described.

# 11 Re-design of a Diagnostic Reasoning System

In this chapter the applicability of the re-design model for compositional systems, presented in Chapters 7, 8, and 9, is illustrated for the re-design of a compositional knowledge-based system for diagnostic reasoning. The trace presented in this chapter is based on the trace presented in (Brazier, Langen, Treur and Wijngaards, 1996). Re-design is performed by a re-design support system in close interaction with a knowledge engineer. The re-design process is explained by reference to a trace, showing the sequence in which components of the model for re-design of compositional systems are activated and the results of these components.

The design object to be re-designed is a simple system for diagnostic reasoning, as shown in Figure 11.1, based on the generic model of diagnosis described in Section 3.3.1 and in Chapter 4. In Figure 11.1 (a), (b) and (c), the different levels of process abstraction described in Chapter 4 are shown.

The requirement qualification set on which the design of the original diagnostic reasoning system was based is shown in Table 11.1: eight requirements and their qualifications. See Section 5.2 for a description of properties and the refinement knowledge with which the refined requirements can be produced.

| *Requirement* | | *Qualification* |
|---|---|---|
| *Identifier* | *Property* | |
| RQa | "The system is capable of diagnostic reasoning." | hard |
| RQb | "The system is capable of initial observations." | hard |
| RQa1 | "The system is capable of determination of hypotheses." | hard |
| RQa2 | "The system is capable of validation of hypotheses." | hard |
| RQa3 | "The system is capable of combining hypothesis determination and hypothesis validation." | hard |
| RQa2.1 | "The system is capable of determination of observations." | hard |
| RQa2.2 | "The system is capable of evaluation of hypotheses." | hard |
| RQa2.3 | "The system is capable of combining observation determination and hypothesis evaluation." | hard |

**Table 11.1** Requirement qualifications for the original diagnostic reasoning system.

One new, additional requirement imposed by the knowledge engineer on the diagnostic reasoning system is shown in Table 11.2.

| *Identifier* | *Property* | *Qualification* |
|---|---|---|
| RQ1 | "The system proposes fewer hypotheses, in comparison to random proposal." | hard |

**Table 11.2** Additional qualified requirement for the diagnostic reasoning system.

First an overview is given of the re-design of this diagnostic reasoning system in Section 11.1. Second, a trace is given of how, in particular, the component Hypothesis Determination is re-designed in Section 11.2. The trace continues in Section 11.3 in which the component Hypothesis Determination is modified according to new requirements. The addition of strategic user interaction is shown in Section 11.4. The approach presented in this chapter is discussed in Section 11.5.

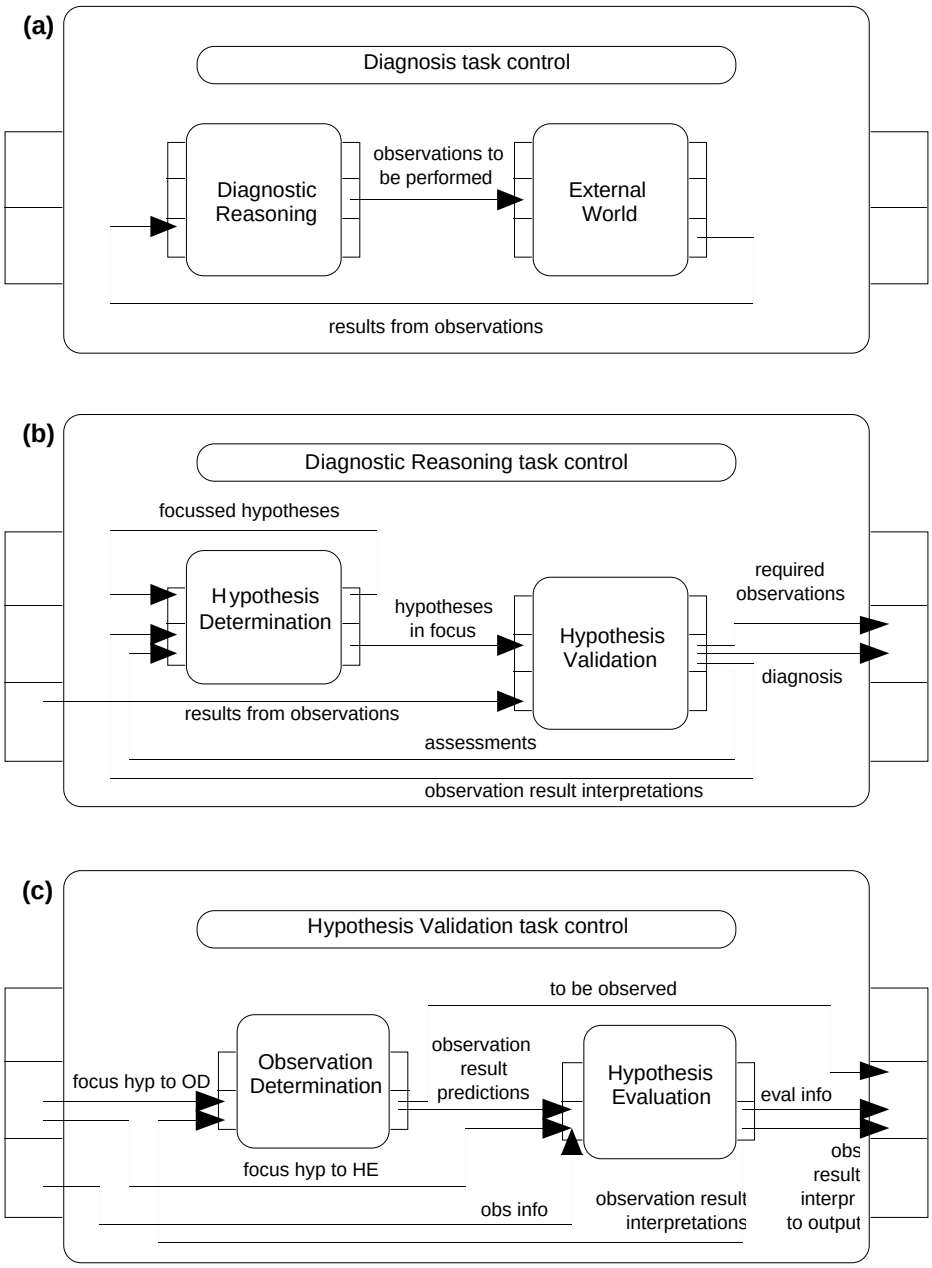


**Figure 11.1** Three levels of process abstraction within the original example diagnostic system

## 11.1 Overview of the re-design process

The re-design of the given diagnostic reasoning system involves several cycles through the re-design model. A brief description of the entire re-design process facilitates the understanding of the traces in the following sections.

The new requirement (added to the original requirements) is shown in Table 11.2, and may or may not affect the design of the diagnostic reasoning system. In this section, first the current design object description is analysed to see if it satisfies the (old and new) requirements. The result of this analysis is that the new requirement is not satisfied. Then the new requirement qualification set is analysed and the new, and rather abstract, qualified requirement RQ1 is to be refined. As a result, new requirements plus qualifications (both ‘hard’ and ‘soft’) emerge, after which the set of qualified requirements is restricted to only the requirements qualified as ‘hard’.

Then the current design object description is analysed to see if it satisfies these requirements. This is not the case: the requirements that resulted from the refinement of RQ1 are not satisfied. To resolve this problem, a number of modifications to the design object description are made, resulting in a new design of the component Hypothesis Determination.

Having succeeded in satisfying the hard requirements, requirements with other qualifications are considered: the soft requirements imposed on the design object description. To satisfy these requirements the design object description is modified: a sub-component Strategy Determination is added to the design of the component Hypothesis Determination together with the appropriate information links and task control.

The knowledge engineer notices that all requirements are satisfied (because s/he was not sure beforehand that they were consistent) and accepts the changes made to the design object description. The knowledge engineer continues the re-design process and adds a requirement qualified as 'hard': the strategy for determination of hypotheses is to be established by the diagnostic reasoning system in interaction with the user. This makes a third round of design object description manipulation necessary, resulting in the decomposition of the sub-component Strategy Determination. After this, the knowledge engineer accepts the new diagnostic reasoning system and imposes no further requirements.

This re-design process is presented in traces in this chapter, showing the activation of (sub-)components chronologically, together with the results of activation. The abbreviations used are listed in Table 11.3.

| *Abbreviation* | *Explanation* |
|---|---|
| KE | knowledge engineer |
| R*n* | requirement *n* |
| RQ*n* | qualified requirement *n* |
| RQS*n* | requirement qualification set *n* |
| DOD*n* | design object description *n* |
| D | Diagnosis (component) |
| EW | External World (component) |
| DR | Diagnostic Reasoning (component) |
| HD | Hypothesis Determination (component) |
| HV | Hypothesis Validation (component) |
| HE | Hypotheses Evaluation (component) |
| OD | Observation Determination (component) |

**Table 11.3** Abbreviations used in the trace.

A compositional system can be described by using the notational convention to have a ':' denote a composition relationship, i.e., D:EW is an abbreviation of 'component EW is a sub-component of component D'. As no naming conflicts occur in this trace, it suffices to use the abbreviated names of components in the composition relation, resulting in the following description of the initial diagnostic reasoning system:

**Characterisation of components in initial diagnostic reasoning system**

The initial diagnostic reasoning system is the first design object description, as defined below:

$DOD_0$={ is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ), is_component( HD ), is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HD, DR:HV, HV:HE, HV:OD }

During modification of a requirement qualification set (or a design object description), first the current requirement qualification set (or design object description) is analysed, after which an explicit focus is made to guide the modification process. A modification is then determined and applied to the current requirement qualification set (or design object description). The result of a modification is taken to be the union of (1) the part that is *not* in focus and (2) the result of the modified focus. After each modification a rationale is constructed. In this rationale, information is stored which keeps track of the specific transition (was a requirement qualification set modified or was a design object description modified?) as well as annotations describing *how* the transition was achieved (which modification method was applied?).

## 11.2 Reducing the number of focussed hypotheses

The knowledge engineer has started the design process co-ordination and indicated that s/he wants to manipulate requirements and their qualifications.

**RQS update of current description**

By adding RQ1 (by the KE) the current description is updated to

{ RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1 }.

**RQS update of modification history**

The history is updated to

$RQS_0$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, Rqa.2.3 }

$RQS_1$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1 }

has_rationale( $\langle RQS_0, DOD_0 \rangle$, $\langle RQS_1, DOD_0 \rangle$, method(KE)).

As it is currently unknown whether the newly added requirement is satisfied by the current DOD, design process co-ordination decides to analyse the current DOD.

**Design process co-ordination**

The current DOD is to be analysed (not modified) on the basis of all requirements in the current RQS.

**Overview of current requirements**

The current requirements taken into account by DODM are:

Ra: “The system is capable of diagnostic reasoning.”

Rb: “The system is capable of initial observations.”

Ra.1: “The system is capable of determination of hypotheses.”

Ra.2: “The system is capable of validation of hypotheses.”

Ra.3: “The system is capable of combining hypothesis determination and hypothesis validation.”

Ra.2.1: “The system is capable of determination of observations.”

Ra.2.2: “The system is capable of evaluation of hypotheses.”

Ra.2.3: “The system is capable of combining observation determination and hypothesis evaluation.”

R1: “The system proposes fewer hypotheses, in comparison to random proposal.”

The current DOD is analysed and is shown *not* to satisfy all requirements. To be more precise, the requirement R1 is not satisfied, the other requirements are satisfied.

**DOD modification**

*1. analysis of current description*

The current DOD does not fulfil all current requirements.

The analysis of the current DOD, given current requirements by DOD manipulation, has finished. Design process co-ordination comes into action, and decides that the current RQS is to be modified.

**Design process co-ordination**

The current RQS is to be modified.

$RQS_1$ is analysed and RQ1 is found to be too abstract: more specific requirement qualifications are needed. This problem is resolved by adding more specific requirement qualifications RQ1.a and RQ1.b on the basis of default reasoning.

**RQS modification**

*1. analysis of current description*

RQ1 is too abstract and RQ1 is stronger than RQa.1.

*2. modification focus determination*

The local focus of modification is set to { RQ1 }.

*3. modification method determination*

The method chosen is *modification by default reasoning*.

*4. modification according to method*

The default requirements and their qualifications are:

RQ1.a: “The system is capable of determination of hypotheses in a structured manner.” (hard)

RQ1.b: “The system is capable of determination of strategies for hypothesis determination.” (soft)

**RQS update of current description**

The current description is updated to

{ RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.b }
by adding RQ1.a and RQ1.b.

**RQS update of modification history**

The history is updated by adding

$RQS_2$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.b }

has_rationale(⟨$RQS_1$, $DOD_0$⟩, ⟨$RQS_2$, $DOD_0$⟩, method( default_reasoning ) )

has_rationale(⟨$RQS_1$, $DOD_0$⟩, ⟨$RQS_2$, $DOD_0$⟩, has_interpretation( RQ1, { RQ1.a, RQ1.b } ) )

has_rationale(⟨$RQS_1$, $DOD_0$⟩, ⟨$RQS_2$, $DOD_0$⟩, strengthens( RQ1, RQa.1 ) )

An example of knowledge used for default reasoning is given below.

**Example of default reasoning knowledge**

The format of the knowledge element is as follows: the first condition specifies which requirement has been selected to be refined by default reasoning. The second condition specifies that the required property has to be present in the current set of qualified requirements, and the conclusion provides possible additions to the current RQS. Note that the conclusions of the second qualified requirement has a specific qualification: ‘soft’. Regardless of the qualification of the requirement in focus, the qualification ‘soft’ is given to the requirement on determining strategies.

```
if   is_qualified_requirement_selected_as_focus( QR: qualified_requirement_name )
  and   holds( is_qualified_requirement(  QR: qualified_requirement_name,
                                          Q: qualification,
                                          R: requirement_name ),
               pos )
  and   holds( is_requirement(            R: requirement_name,
                                          has_property( S: system_name,
                                                proposing_fewer_hypotheses_compared_to_
                                                random_proposal ) ),
               pos )
then    addition_to_current_RQS(
            is_qualified_requirement(   new_name( QR: qualified_requirement_name, a ),
                                        Q: qualification,
                                        new_name( R: requirement_name, a ) )
  and   addition_to_current_RQS(
            is_requirement(             new_name( R: requirement_name, a ),
                                        has_property( S: system_name,
                                                structured_determination_of_hypotheses ) )
```

```
and   addition_to_current_RQS(
          is_qualified_requirement(   new_name( QR : qualified_requirement_name, b ),
                                      soft,
                                      new_name( R: requirement_name, b ) )
and   addition_to_current_RQS(
          is_requirement(             new_name( R: requirement_name, b ),
                                      has_property( S: system_name,
                                            is_capable_of_determination_of_strategies_
                                                  for_hypothesis_determination ) );
```

$RQS_2$ is analysed and further refined in a unique manner: domain specific knowledge on requirements is available to infer (in a deductive manner) more specific, requirement qualifications from RQ1.a.

**RQS modification**

*1. analysis of current description*

RQ1.a is too abstract and is qualified as 'hard'; RQ1.b is too abstract and is qualified as 'soft'. The qualification 'hard' is given precedence over the qualification 'soft'.

*2. modification focus determination*

The focus of modification is set to { RQ1.a }.

*3. modification method determination*

The method chosen is *modification by deductive refinement*.

**RQS deductive refinement**

The newly proposed requirements and their qualifications are:

RQ1.a.1: "The system is capable of generation of hypotheses." (hard)

RQ1.a.2: "The system is capable of comparison of hypotheses." (hard)

RQ1.a.3: "The system is capable of selection of hypotheses." (hard).

RQ1.a.4: "The system is capable of combining generation of hypotheses, comparison of hypotheses, and selection of hypotheses." (hard).

**RQS update of current description**

The current description is updated to
{ RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.b, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4 }.

**RQS update of modification history**

The history is updated by adding

$RQS_3$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.b, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4 }

has_rationale($\langle RQS_2, DOD_0 \rangle$, $\langle RQS_3, DOD_0 \rangle$, method( deductive_refinement ) )

has_rationale($\langle RQS_2, DOD_0 \rangle$, $\langle RQS_3, DOD_0 \rangle$, has_deductive_refinement( RQ1.a, { RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4 } ).

The deductive refinement of RQ1.a into four more specific requirements is shown in the knowledge below. The inverse, the conjunction of the specific requirements implies the requirement RQ1.a, also holds, and this relation on these properties is employed in deductive DOD refinement.

**Example knowledge for deductive refinement of a requirement**

The sample knowledge element below illustrates specific deductive refinement knowledge on requirements. The first condition of the knowledge element specifies a qualified requirement, which refers to a specific unqualified requirement. The second condition specifies that a specific expression is related to that requirement. The conclusions specify four (refined) qualified requirements and requirements that are

refinements of the requirement specified in the conditions. All of the refined qualified requirements have the same qualification as the qualified requirement in the condition part.

```
if   is_qualified_requirement(          QR: qualified_requirement_name,
                                        Q: qualification,
                                        R: requirement_name )
     and   is_requirement(              R: requirement_name,
                                        has_property( S: system_name,
                                                is_capable_of_structured_
                                                      determination_of_hypotheses ) )
then      is_qualified_requirement(     new_name( QR: qualified_requirement_name,.1 ),
                                        Q: qualification,
                                        new_name( R: requirement_name,.1 ) )
     and  is_qualified_requirement(     new_name( QR: qualified_requirement_name,.2 ),
                                        Q: qualification,
                                        new_name( R: requirement_name,.2 ) )
     and  is_qualified_requirement(     new_name( QR: qualified_requirement_name,.3 ),
                                        Q: qualification,
                                        new_name( R: requirement_name,.3 ) )
     and  is_qualified_requirement(     new_name( QR: qualified_requirement_name,.4 ),
                                        Q: qualification,
                                        new_name( R: requirement_name,.4 ) )
     and  is_requirement(               new_name( R: requirement_name,.1 ),
                                        has_property( S: system_name,
                                              is_capable_of_generation_of_hypotheses ) )
      and is_requirement(               new_name( R: requirement_name,.2 ),
                                        has_property( S: system_name,
                                              is_capable_of_comparison_of_hypotheses ) )
     and  is_requirement(               new_name( R: requirement_name,.3 ),
                                        has_property( S: system_name,
                                              is_capable_of_selection_of_hypotheses ) )
     and  is_requirement(               new_name( R: requirement_name, .4 ),
                                        has_property( S: system_name,
                                              is_capable_of_combining_generation_and_
                                              comparison_and_selection_of_hypotheses ) );
```

$RQS_3$ is analysed and there seem to be no more problems. However, whether the initial DOD satisfies the new requirements is unknown: no modification has been made to the design object description since the introduction of the new requirement qualifications; the current DOD has not even been checked (as concluded from inspection of the history: the $DOD_0$ is the only DOD that is known). The strategy chosen is to initially disregard all requirements with lower qualifications than ‘hard’.

**RQS modification**

*1. analysis of current description*

There seem to be no more problems with the current RQS, but, since no attempt has been made to make a new DOD since the introduction of new requirement qualifications, all requirements with non-hard qualifications are disregarded for the moment.

*2. modification focus determination*

The focus of modification is set to { RQ1.b }.

*3. modification method determination*

The method chosen is *modification by deletion*.

*4. modification according to method*

All requirement qualifications in focus are deleted.

**RQS update of current description**

The current description is updated to
{ RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4 }.

**RQS update of modification history**

The history is updated by adding

$RQS_4$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4 }

has_rationale($\langle RQS_3, DOD_0 \rangle$, $\langle RQS_4, DOD_0 \rangle$, method( deletion ) )

has_rationale($\langle RQS_3, DOD_0 \rangle$, $\langle RQS_4, DOD_0 \rangle$, has_lower_qualification_than( { RQ1.b }, hard ).

After analysis that no further RQS manipulation is needed, design process co-ordination comes into action and decides, on the basis of information of the histories of the current RQS and the current DOD, that the current DOD is to be analysed and possibly modified.

**Design process co-ordination**

The current DOD is to be analysed and possibly modified, on the basis of all requirements in the current RQS.

**Overview of current requirements**

The current requirements taken into account by DODM are:

Ra: “The system is capable of diagnostic reasoning.”

Rb: “The system is capable of initial observations.”

Ra.1: “The system is capable of determination of hypotheses.”

Ra.2: “The system is capable of validation of hypotheses.”

Ra.3: “The system is capable of composition of hypothesis determination and hypothesis validation.”

Ra.2.1: “The system is capable of determination of observations.”

Ra.2.2: “The system is capable of evaluation of hypotheses.”

Ra.2.3: “The system is capable of composition of observation determination and hypothesis evaluation.”

R1: “The system proposes fewer hypotheses, in comparison to random proposal.”

R1.a: “The system is capable of determining hypotheses in a structured manner.”

R1.a.1: “The system is capable of generation of hypotheses.”

R1.a.2: “The system is capable of comparison of hypotheses.”

R1.a.3: “The system is capable of selection of hypotheses.”

R1.a.4: “The system is capable of combing generation of hypotheses, comparison of hypotheses, and selection of hypotheses”.

$DOD_0$, the description of the original diagnostic reasoning system with components D, EW, DR, HD, HV, HE, and OD, and sub-component relations D:EW, D:DR, DR:HD, DR:HV, HV:HE, and HV:OD, is analysed and is shown not to satisfy all current requirements. In particular, the specification of the component HD (meant originally to fulfil Ra.1) does not fulfil the requirements R1, R1.a, R1.a.1, R1.a.2, R1.a.3, and R1.a.4 (notice that R1.a.1, R1.a.2, R1.a.3, and R1.a.4 are meant to refine R1.a, which is a default interpretation of R1, which is stronger than Ra.1, according to the history). A possible solution to resolve this problem is to replace HD by a component taken from the library.

**DOD modification**

*1. analysis of current description*

The current DOD does not fulfil all current requirements.

*2. modification focus determination*

The focus of modification is set to { HD }.

*3. modification method determination*

The method chosen is *modification based on library consultation.*

*4. modification according to method*

HD is replaced by a composed component capable of structured determination of hypotheses (*libStructD*), with generic sub-components for generation (*libG*), comparison (*libC*) and selection (*libS*), which are all renamed to refer to the context of the hypothesis determination task.

**DOD update of current description**

The current description is updated to
{ is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ), is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HV, HV:HE, HV:OD, is_component( HD* ), is_component( HG ), is_component( HC ), is_component( HS ), DR:HD*, HD*:HG, HD*:HC, HD*:HS }.

**DOD update of modification history**

The history is updated to

$DOD_0$={ is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ), is_component( HD ), is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HD, DR:HV, HV:HE, HV:OD }

$DOD_1$ = { is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ), is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HV, HV:HE, HV:OD, is_component( HD* ), is_component( HG ), is_component( HC ), is_component( HS ), DR:HD*, HD*:HG, HD*:HC, HD*:HS }

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, replaced_by( HD, HD*) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, meant_to_satisfy( HD*, { R1.a } ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, meant_to_satisfy( HG, { R1.a.1 } ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, meant_to_satisfy( HC, { R1.a.2 } ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, meant_to_satisfy( HS, { R1.a.3 } ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, meant_to_satisfy( { HD*, HG, HC, HS }, { R1.a.4 } ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, method( library_consultation ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, is_based_on( HD*, *libStructD* ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, is_based_on( HG, *libG* ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, is_based_on( HC, *libC* ) )

has_rationale(⟨$RQS_4$, $DOD_0$⟩, ⟨$RQS_4$, $DOD_1$⟩, is_based_on( HS, *libS* ) ).

$DOD_1$ is analysed and $DOD_1$ still contains components that are generic and need domain knowledge to perform their task in the domain of application. It is the knowledge engineer's job to provide this domain knowledge.

**DOD modification**

*1. analysis of current description*

The components HD*, HG, HC, HS are not instantiated; thus, the current DOD is not complete.

*2. modification focus determination*

The focus of modification is set to { HD*, HG, HC, HS }.

*3. modification method determination*

The method chosen is *modification by the KE*.

*4. modification according to method*

The refinements added to the description by the KE are:

$HG_{inst}$, which is the instantiation of HG with domain-specific knowledge,

$HC_{inst}$, which is the instantiation of HC with domain-specific knowledge,

$HS_{inst}$, which is the instantiation of HS with domain-specific knowledge.

**DOD update of current description**

The current description is updated to
{ is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ),
is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HV, HV:HE, HV:OD,
is_component( HD* ), is_component( $HG_{inst}$ ), is_component( $HC_{inst}$),
is_component( $HS_{inst}$ ), DR:HD*, HD*:$HG_{inst}$, HD*:$HC_{inst}$, HD*:$HS_{inst}$ }.

**DOD update of modification history**

The history is updated by adding

$DOD_2$ = { is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ),
is_component( HV ), is_component( HE ), is_component( OD ),
D:EW, D:DR, DR:HV, HV:HE, HV:OD,
is_component( HD* ), is_component( $HG_{inst}$ ), is_component( $HC_{inst}$ ),
is_component( $HS_{inst}$ ), DR:HD*, HD*:$HG_{inst}$, HD*:$HC_{inst}$, HD*:$HS_{inst}$ }

has_rationale(⟨$RQS_4$, $DOD_1$⟩, ⟨$RQS_4$, $DOD_2$⟩, method( KE ) )

has_rationale(⟨$RQS_4$, $DOD_1$⟩, ⟨$RQS_4$, $DOD_2$⟩, is_instantiation_of( $HG_{inst}$, HG ) )

has_rationale(⟨$RQS_4$, $DOD_1$⟩, ⟨$RQS_4$, $DOD_2$⟩, is_instantiation_of( $HC_{inst}$, HC ) )

has_rationale(⟨$RQS_4$, $DOD_1$⟩, ⟨$RQS_4$, $DOD_2$⟩, is_instantiation_of( $HS_{inst}$, HS ) ).

$DOD_2$ is analysed. The conclusion is that manipulation of the current DOD has been successfully accomplished. Therefore, design process co-ordination becomes active and decides, on the basis of information of the histories of the current RQS and the current DOD, that the current RQS may be further analysed and modified if necessary.

**DOD modification**

*1. analysis of current description*

The current description fulfils all requirements and all components are instantiated.

Below an example is given of how a specific property can be identified in the example diagnostic system.

**Example knowledge for deductive refinement of a design object description**

The conditions of the knowledge below specify that if a component C (characterised as being designed for structured determination of some information concept I) has a sub-component D (characterised as being designed for the generation of information concept I), and the component D has an information type in its output interface which contains instantiated information designed for the representation of the generation of information concept I, and the component D has a knowledge base which contains instantiated knowledge designed for the generation of information concept I and information concept I is 'hypotheses'. Then the component C can be said to have the property is capable of generation of hypotheses.

```
if  is_component(                          C: component_name )
    and   has_characterisation(            C: component_name,
                                           structured_determination_of( I: information_concept ) )
    and   is_component (                   D: component_name )
    and   has_subcomponent(                C: component_name,
                                           D: component_name )
    and   has_characterisation(            D: component_name,
                                           generation_of( I: information_concept ) )
    and   has_interface_information_type(  D: component_name,
                                           output_interface,
                                           Dout: information_type_name )
    and   has_characterisation(            Dout: information_type_name,
                                           generation_of( I: information_concept ) )
```

```
    and   has_knowledge_base(     D: component_name,
                                  K: knowledge_base_name )
    and   has_characterisation(   K: knowledge_base_name,
                                  generation_of( I: information_concept ) )
    and   has_characterisation(   K: knowledge_base_name,
                                  contains_instantiated_knowledge )
    and   I: information_concept = hypotheses
  then    has_property(           C: component_name,
                                  is_capable_of_generation_of_hypotheses );
```

The re-design process continues with an analysis of what its next actions are going to be.

**design process co-ordination**

The current RQS is to be modified.

The first results of re-designing the diagnostic reasoning system are shown in Figure 11.2.

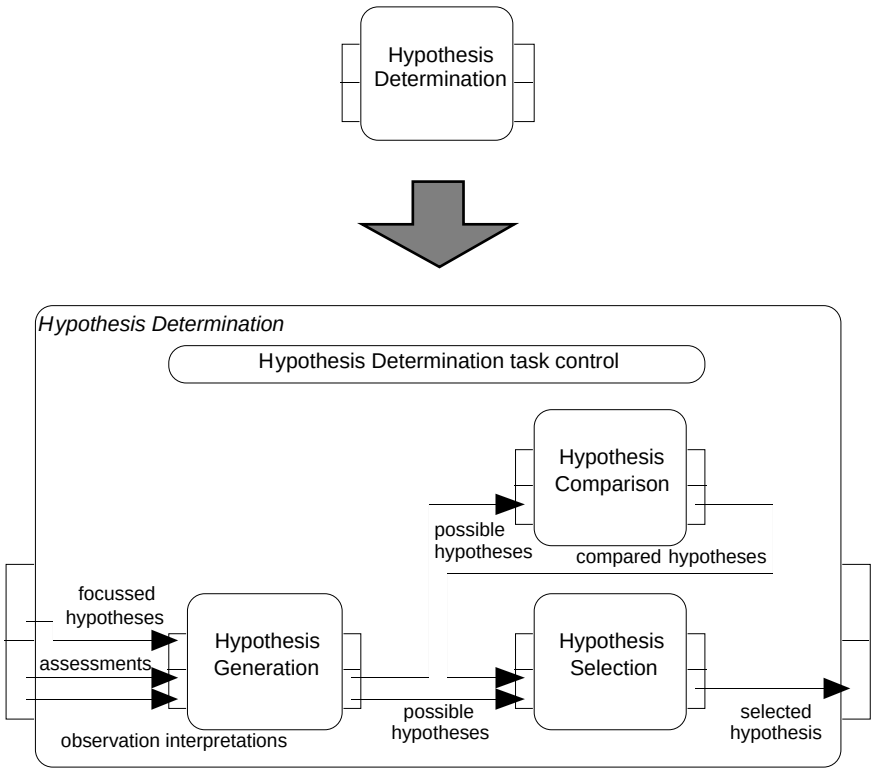


**Figure 11.2** Results of the first change to the diagnostic reasoning system: from description $DOD_0$ to description $DOD_2$.

## 11.3 Integration of hypotheses determination strategies

$RQS_4$ is analysed and there seem to be no problems. However, the requirements with non hard qualifications have not yet been considered; this is concluded from inspecting the history. As a result, the decision to analyse the current DOD with respect to the non hard requirements is made. (Note that from this point on, the trace is continued in an abbreviated form. The global results of only the components for modification, deductive refinement, update of modification history, overview of current requirements, and design process co-ordination are presented.)

**RQS modification**

*1. analysis of current description*

There seem to be no problems with the current RQS, but requirements with non hard qualifications have not yet been considered.

*2. modification focus determination*

The focus of modification is set to { }.

*3. modification method determination*

The method chosen is *modification by retrieval from history*.

*4. modification according to method*

All requirements with lower qualifications than hard, that have been deleted from the requirement qualification set in the past according to history, are identified: in this case, RQ1.b.

**RQS update of current description**

The current description is updated to
{ RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4, RQ1.b }.

**RQS update of modification history**

The history is updated by adding

$RQS_5$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4, RQ1.b }

has_rationale(⟨$RQS_4$, $DOD_2$⟩, ⟨$RQS_5$, $DOD_2$⟩, method( retrieval_from_history ) )

has_rationale(⟨$RQS_4$, $DOD_2$⟩, ⟨$RQS_5$, $DOD_2$⟩, has_lower_qualification_than({ RQ1.b }, hard ).

After RQS manipulation has been completed, design process co-ordination comes into action and decides that the current DOD has to be analysed and possibly modified.

**Design process co-ordination**

The current DOD has to be analysed and possibly modified, on the basis of the current RQS.

**Overview of current requirements**

The set of current requirements taken into account by DODM is extended with:

R1.b: “The system is capable of determination of strategies for hypothesis determination.”

**DOD modification**

$DOD_2$ is analysed. It does not satisfy all requirements. In particular, the component HD* does not fulfil requirement R1.b. Therefore, HD* is *modified by means of library consultation*, so as to include a new generic sub-component for strategy determination, StratD, which is based on the library component *libStratD*.

**DOD update of modification history**

The history is updated by adding

$DOD_3$ = { is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ), is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HV, HV:HE, HV:OD, is_component( HD* ), is_component( $HG_{inst}$ ), is_component( $HC_{inst}$ ), is_component( $HS_{inst}$ ), DR:HD*, HD*:$HG_{inst}$, HD*:$HC_{inst}$, HD*:$HS_{inst}$ , is_component( StratD ), HD*:StratD }.

**DOD modification**

$DOD_3$ is analysed. The component StratD needs domain knowledge to perform its task in the domain of application. This generic component is instantiated by means of *modification by the KE*.

**DOD update of modification history**

The history is updated by adding

$DOD_4$ = { is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ),
is_component( HV ), is_component( HE ), is_component( OD ),
D:EW, D:DR, DR:HV, HV:HE, HV:OD,
is_component( HD* ), is_component( $HG_{inst}$ ), is_component( $HC_{inst}$ ),
is_component( $HS_{inst}$ ), DR:HD*, HD*:$HG_{inst}$, HD*:$HC_{inst}$, HD*:$HS_{inst}$ ,
is_component( $StratD_{inst}$ ), HD*:$StratD_{inst}$ }.

**DOD modification**

$DOD_4$ is analysed: it fulfils all current requirements and all components are instantiated.

**design process co-ordination**

Manipulation of the current DOD has been successfully accomplished: the current DOD satisfies all current requirements. The current RQS is to be modified.

The results of the second change to the diagnostic reasoning system are shown in Figure 11.3.

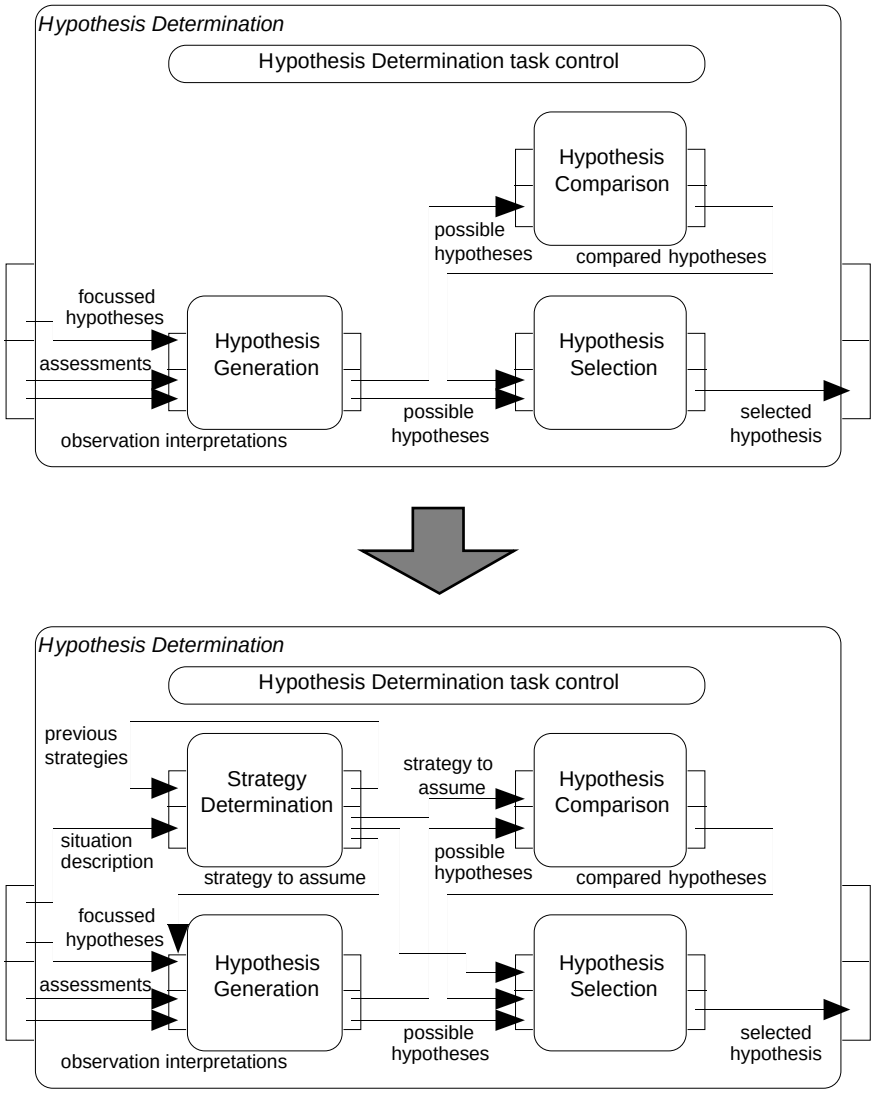


**Figure 11.3** Results of the second change to the diagnostic reasoning system: from description $DOD_2$ to description $DOD_4$.

## 11.4 Realisation of strategic user interaction for determination of hypotheses

The next steps analyse and possibly modify the current RQS, on the grounds that the changes to the diagnostic reasoning system satisfy the current requirements.

**RQS modification**

Now all requirements, hard and soft, have been satisfied, the KE is asked whether any further modifications to the current RQS are needed. The result of *modification by the KE* is the addition of the following single requirement plus qualification:

RQ2: "The system determines strategies on the basis of interaction with the user." (hard).

Furthermore, the qualified requirement RQ2 strengthens the qualified requirement RQ1.b.

**RQS update of modification history**

The history is updated by adding

$RQS_6$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4, RQ1.b, RQ2 }.

**RQS modification**

The qualified requirement RQ2 can be refined by employing domain specific knowledge on requirements. The method chosen is *modification by deductive refinement*, which results in the following four more specific qualified requirements:

RQ2.a: "The system is capable of proposing strategies." (hard).

RQ2.b: "The system is capable of acquisition of user feedback." (hard).

RQ2.c: "The system is capable of selection of strategies." (hard).

RQ2.d: "The system is capable of combining proposing strategies, acquisition of user feedback, and selection of strategies." (hard).

**RQS update of modification history**

The history is updated by adding

$RQS_7$ = { RQa, RQb, RQa.1, RQa.2, RQa.3, RQa.2.1, RQa.2.2, RQa.2.3, RQ1, RQ1.a, RQ1.a.1, RQ1.a.2, RQ1.a.3, RQ1.a.4, RQ1.b, RQ2, RQ2.a, RQ2.b, RQ2.c, RQ2.d }.

**RQS modification**

There seem to be no more problems in $RQS_7$.

**design process co-ordination**

Manipulation of the current RQS has been successfully accomplished, so therefore the current DOD is to be manipulated, on the basis of all requirements in the current RQS.

**Overview of current requirements**

The set of current requirements is extended with:

R2: "The system determines strategies on the basis of interaction with the user."

R2.a: "The system is capable of proposing strategies."

R2.b: "The system is capable of acquisition of user feedback."

R2.c: "The system is capable of selection of strategies."

R2.d: "The system is capable of combining proposing strategies, acquisition of user feedback, and selection of strategies."

**DOD modification**

The component $StratD_{inst}$ specified in $DOD_4$ (meant to satisfy R1.b) does not fulfil requirement R2 (which strengthens R1.b). Therefore, $StratD_{inst}$ is replaced, based on *modification by library consultation*, by a composed component capable of interactive strategy determination StratD* (which is based on the library

component *libInterStratD*). This composed component contains three sub-components, one for the system to propose strategies, one for the user (to give feedback on proposed strategies) and one for selecting a strategy to use: StratProp, StratFeedback, and StratSelect (which are based on the library components *libStratProp*, *libStratFeedback*, and *libStratSelect*). The co-operative aspect of the system and the user lies in the user giving feedback on proposed strategies, which results in either a strategy being selected, or again a new round of proposing strategies and the user giving feedback on the proposed strategies.

**DOD update of modification history**

The history is updated by adding

DOD5 = { is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ), is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HV, HV:HE, HV:OD, is_component( HD* ), is_component( $HG_{inst}$ ), is_component( $HC_{inst}$ ), is_component( $HS_{inst}$ ), DR:HD*, HD*:$HG_{inst}$, HD*:$HC_{inst}$, HD*:$HS_{inst}$ , is_component( StratD* ), is_component( StratProp ), is_component( StratFeedback ), is_component( StratSelect ), HD*:StratD*, StratD*:StratProp, StratD*:StratFeedback, StratD*:StratSelect }.

**DOD modification**

$DOD_5$ is analysed. The sub-components of StratD* need domain knowledge to perform their task in the domain of application: these generic components are instantiated by means of *modification by the KE*.

**DOD update of modification history**

The history is updated by adding

$DOD_6$ = { is_top_level( D ), is_component( D ), is_component( EW ), is_component( DR ), is_component( HV ), is_component( HE ), is_component( OD ), D:EW, D:DR, DR:HV, HV:HE, HV:OD, is_component( HD* ), is_component( $HG_{inst}$ ), is_component( $HC_{inst}$ ), is_component( $HS_{inst}$ ), DR:HD*, HD*:$HG_{inst}$, HD*:$HC_{inst}$, HD*:$HS_{inst}$ , is_component( StratD* ), is_component( $StratProp_{inst}$ ), is_component( $StratFeedback_{inst}$ ), is_component( $StratSelect_{inst}$ ), HD*:StratD*, StratD*:$StratProp_{inst}$, StratD*:$StratFeedback_{inst}$, StratD*:$StratSelect_{inst}$ }.

**DOD modification**

$DOD_6$ is analysed: it fulfils all current requirements and all components are instantiated.

**design process co-ordination**

Manipulation of the current DOD has been successfully accomplished: the current DOD satisfies all requirements in the current RQS. Therefore, the current RQS is to be modified.

The third change to the diagnostic reasoning system is shown in Figure 11.4.

Even though all requirements stated by the KE are satisfied, s/he may again change his/her mind about, or have new ideas regarding, the structure and the behaviour of the compositional architecture for diagnostic reasoning.

**RQS modification**

The KE makes no changes to $RQS_7$.

**design process co-ordination**

Manipulation of the current RQS has been successfully accomplished, no alterations were made to the current RQS, and the current DOD satisfies all requirements of the current RQS. The design process comes to an end.

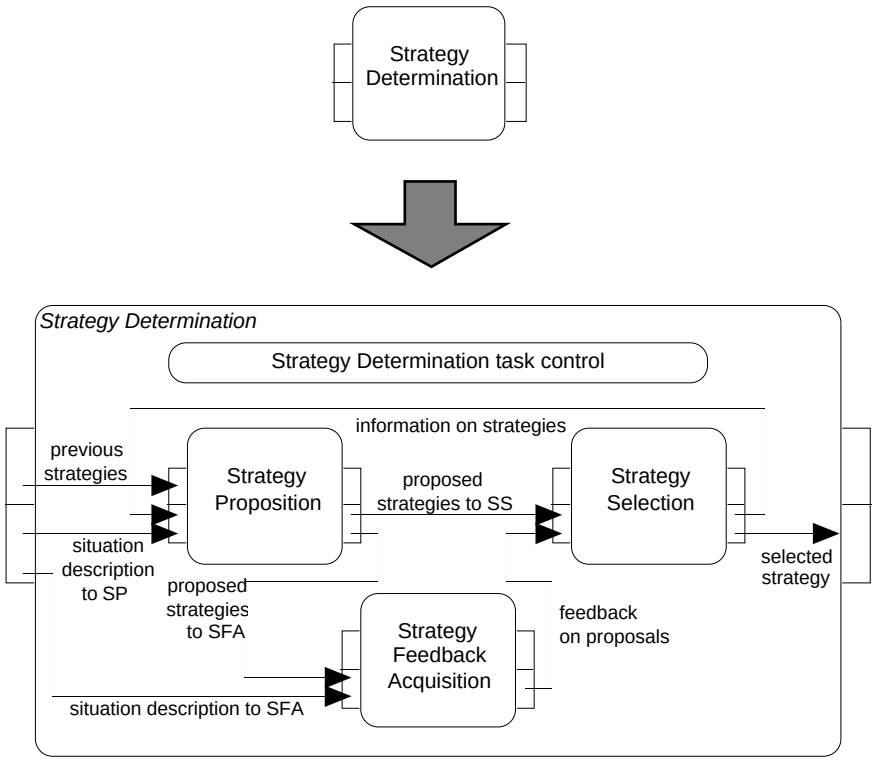


**Figure 11.4** Third change to the diagnostic reasoning system: from description $DOD_4$ to description $DOD_6$.

## 11.5 Discussion

In this chapter an application of the model for re-design of compositional systems has been shown: an example diagnostic reasoning system has been re-designed. Within the trace presented in this chapter all three main processes within design have been shown to be important: the manipulation of qualified requirements, manipulation of design object descriptions and explicit control over the re-design process. For the re-design of this diagnostic reasoning system the realisations of all desiderata (see Section 2.1 and Section 2.2) have been employed.

In the trace, examples have been given of the application of

- deductive refinement knowledge on design requirements,
- a rationale for transitions between sets of qualified requirements,
- modification of a design object description by means of re-using a model from a library,
- deduction of properties of a design object description to assess requirements,
- a rationale for transitions between design object descriptions,
- explicit control over the process of re-design, and
- alternating between the manipulation of sets of qualified requirements and the manipulation of design object descriptions.

A number of requirements inducing refinement of requirements and replacement of components by compositions of other components are described.

Another example of the re-design of a compositional structure is presented in Chapter 12.

# 12 Re-design of a Multi-Agent System

Two desiderata on re-design of a compositional system have not yet been addressed: desiderata dr6 (“model of integration of design and realisation”) and dr7 (“model of self-modification”). Desideratum dr6 states that the results of a design process (including a design object description) need to be integrated with the realisation of an artefact, according to the description of the design object. Desideratum dr7 describes that the object of design is a description of the system in which the re-design process is specified as well.

In this chapter a *self-modifying multi-agent system* is described which contains an agent capable of modifying the multi-agent system in which the agent itself plays a role. To be more specific, a design agent (described in Chapter 10) creates and dynamically adds a new agent to the existing multi-agent system. The trace presented is an adapted version of the trace presented in (Brazier, Jonker, Treur and Wijngaards, 1998*a*).

In Section 12.1 an overview of the modification process is given. Section 12.2 describes self-modification by the multi-agent system: the re-design process within the Personal Assistant, including the creation action. Section 12.3 contains the discussion of this chapter.

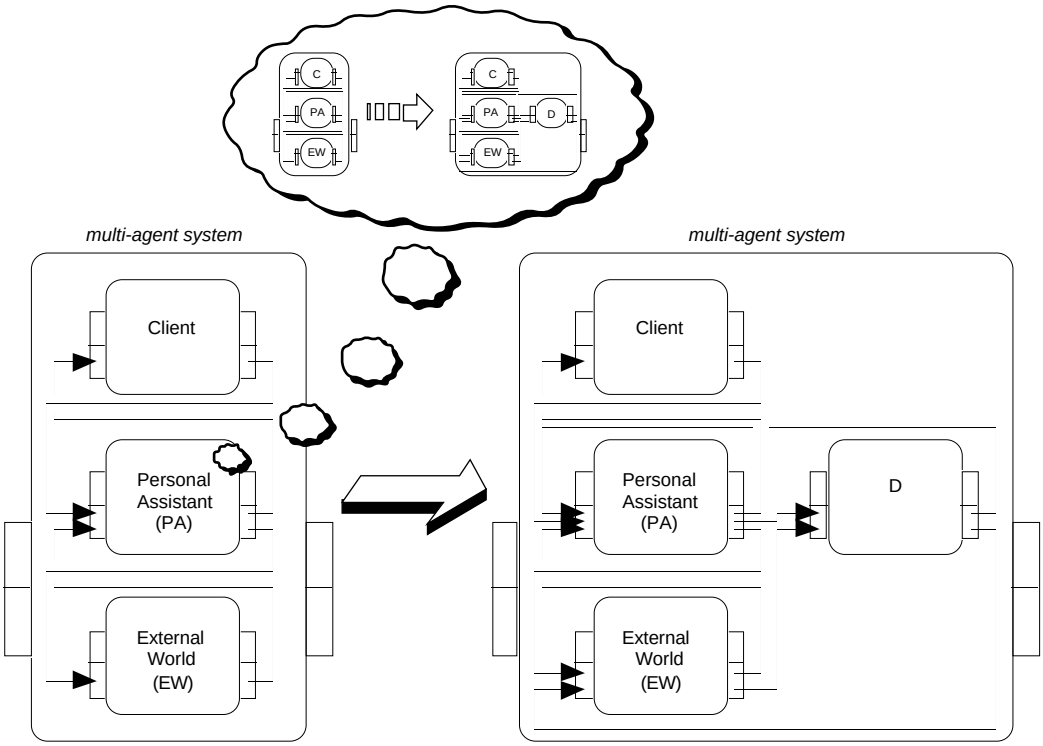


**Figure 12.1** Redesign of a multi-agent system: agent PA modifies the structure of the multi-agent system by creating a new agent D.

## 12.1 Overview of the modification process

The redesign process is illustrated in Figure 12.1. The left box contains the multi-agent system (consisting of the two agents Personal Assistant and Client, and the component External World)

before modification, the right box depicts the multi-agent system after modification (with an additional agent D). The Personal Assistant (PA) is the design agent (See Chapter 10). It first reasons to obtain a plan for the desired modification (the cloud depicts the reasoning process), and executes this plan by performing (material) actions in the External World.

The External World represents all material aspects, including the material aspects of the agents. An operational system (i.e., a 'running' system) has a material representation, which describes the structure of the system. On the basis of the material aspects of the current multi-agent system, the design agent designs a new agent. After a new agent has been designed by the design agent, this design is effectuated by execution of a (material) creation action initiated by the design agent in the external world. After this creation action, the multi-agent system functions with the additional agent.

For self-modification of a system, two elements are of importance: a re-design process which generates modification descriptions of the system, and a mechanism which links descriptions of a system to itself.

The design agent described in Chapter 10 forms the basis for the Personal Assistant agent in the example multi-agent system. This implies that the Personal Assistant agent has the ability to re-design compositional systems. In this example the Personal Assistant agent has to re-design a multi-agent system: a new agent has to be added to the multi-agent system. The Personal Assistant agent is part of the multi-agent system to be re-designed: the Personal Assistant agent could re-design itself as well.

## 12.2 Trace of re-design

A trace of the re-design of the multi-agent system described in the previous section involves an explanation of the following issues:

- the initial situation (Section 12.2.1),
- representation of requirements (Section 12.2.2),
- manipulation of requirements (Section 12.2.3),
- representation of an agent design within a design agent (Section 12.2.4)
- manipulation of design object descriptions (Section 12.2.5), and
- creation action: realisation of a designed agent (Section 12.2.6).

### 12.2.1 Initial Situation

The initial situation, i.e., the structure of the multi-agent system before it is modified, and the requirements on the new multi-agent system are outlined below.

**Situation description**

Figure 12.1 depicts the initial multi-agent system. Agent C represents the (human) client; agent PA represents a Personal Assistant. The Client can ask certain questions and the Personal Assistant provides answers to these questions. For the sake of the example, consider the situation in which the Client requests specific information. The Personal Assistant receives this request, and realizes that it does not have the information requested. However, the Personal Assistant is able to create other agents to solve specific types of problems. To this end a number of requirements need to be formulated, and information on the structure of the multi-agent system needs to be acquired, on the basis of which a new agent can be created to search for information to answer the client's requests.

### 12.2.2 Representation of requirements

Requirements are formulated in terms of abilities and properties of agents and the external world. As described in Chapter 5 abilities and properties can be assigned to

- individual agents,
- the external world,
- an individual agent in relation to the agents and the world with which it interacts,
- the world in relation to the agents with which it interacts, and
- a multi-agent system as a whole.

**Prerequisites for re-design**

The design agent (i.e., agent_A) formulates the following initial requirements for the new agent:

```
is_qualified_requirement( qr_m1, hard, r_m1 );
is_requirement( r_m1, has_property(   mas_S,
                                      is_capable_of_distributed_information_gathering(
                                            agent_A, agent_D, world_W ) ) );

is_qualified_requirement( qr_m2, hard, r_m2 );
is_requirement( r_m2, has_property(   agent_D,
                                      is_capable_of_information_gatherering_for(
                                            agent_A, scientific_publications ) ) );
```

These requirements state that the new agent should solve sub-problems for the existing agent A by gathering information (which takes place in the external world). The new agent's specific subject of expertise is that, it should be capable of gathering information on scientific publications (e.g. on the Internet). To design a new agent, also knowledge of the structure of the existing multi-agent system is needed. To this end the design agent A makes an explicit observation in the external world EW and observes the structure of the existing multi-agent system. The agent's design task commences on the basis of these requirements and the structure of the multi-agent system.

### 12.2.3 Manipulation of requirements

Abilities of agents such as co-operation, bi-directional communication, and world interaction are often needed for agents to jointly be able to perform a certain task. Knowledge on refinements of the ability of bi-directional communication can be formally represented (see Chapter 5: "compositional architectures: properties" and see the example below). Meta-reasoning is employed to decide which refinement alternative should be employed for which ability.

**Representation of requirements refinement knowledge**

Below two knowledge elements are presented that correspond to two refinements shown in Figure 12.2 and refinement relations defined in Figure 5.6. The format of each knowledge element is as follows: the first condition specifies which qualified requirement has been selected to be refined. The second and third condition specify the required property, and the fourth condition indicates which refinement alternative should be considered (concluded elsewhere). The conclusions provide possible refinements for the qualified requirement in focus.

```
if   is_qualified_requirement_selected_as_focus( QR: qualified_requirement_name )
  and   holds( is_qualified_requirement(  QR: qualified_requirement_name,
                                          Q: qualification,
                                          R: requirement_name ),
               pos )
  and   holds( is_requirement(            R: requirement_name,
                                          has_property( A: component_name,
                                                is_capable_of_bidirectional_communication_
                                                with( A2: component_name )) ),
               pos )
  and   refinement_alternative( specialisations )
then    addition_to_current_RQS(
            is_qualified_requirement(     new_name( QR: qualified_requirement_name, a ),
                                          Q: qualification,
                                          new_name( R: requirement_name, a ) ) )
  and   addition_to_current_RQS(
            is_requirement(               new_name( R: requirement_name, a ),
                                          has_property( A: component_name,
                                                is_capable_of_unidirectional_communication_
                                                from( A2: component_name ) ) ) )
  and   addition_to_current_RQS(
            is_qualified_requirement(     new_name( QR : qualified_requirement_name, b ),
                                          Q: qualification,
                                          new_name( R: requirement_name, b ) ) )
```

```
    and   addition_to_current_RQS(
              is_requirement(          new_name( R: requirement_name, b ),
                                       has_property( A: component_name,
                                             is_capable_of_unidirectional_communication_
                                                   to( A2: component_name ) ) ) )
    and   addition_to_current_RQS(
              is_qualified_requirement(  new_name( QR: qualified_requirement_name, c ),
                                       Q: qualification,
                                       new_name( R: requirement_name, c ) ) )
    and   addition_to_current_RQS(
              is_requirement(          new_name( R: requirement_name, c ),
                                       has_property( A: component_name,
                                       is_capable_of_combining_unidirectional_
                                             communication_from_and_to(
                                                   A2: component_name ) ) ) );

if  is_qualified_requirement_selected_as_focus( QR: qualified_requirement_name )
    and   holds( is_qualified_requirement(  QR: qualified_requirement_name,
                                       Q: qualification,
                                       R: requirement_name ),
              pos )
    and   holds( is_requirement(           R: requirement_name,
                                       has_property( A: component_name,
                                             is_capable_of_unidirectional_communication_
                                             from( A2: component_name )) ),
              pos )
    and   refinement_alternative( realisations )
then      addition_to_current_RQS(
              is_qualified_requirement(  new_name( QR: qualified_requirement_name, a ),
                                       Q: qualification,
                                       new_name( R: requirement_name, a ) ) )
    and   addition_to_current_RQS(
              is_requirement(          new_name( R: requirement_name, a ),
                                       has_property( A: component_name,
                                             is_capable_of_reasoning_about_
                                             unidirectional_communication_
                                             from( A2: component_name ) ) ) )
    and   addition_to_current_RQS(
              is_qualified_requirement(  new_name( QR : qualified_requirement_name, b ),
                                       Q: qualification,
                                       new_name( R: requirement_name, b ) ) )
    and   addition_to_current_RQS(
              is_requirement(          new_name( R: requirement_name, b ),
                                       has_property( A: component_name,
                                             is_capable_of_executing_unidirectional_
                                             communication_to( A2: component_name ) ) ) )
    and   addition_to_current_RQS(
              is_qualified_requirement(  new_name( QR: qualified_requirement_name, c ),
                                       Q: qualification,
                                       new_name( R: requirement_name, c ) ) )
    and   addition_to_current_RQS(
              is_requirement(          new_name( R: requirement_name, c ),
                                       has_property( A: component_name,
                                       is_capable_of_combining_reasoning_and_
                                             executing_unidirectional_communication_
                                                   from( A2: component_name ) ) ) );
```

On the basis of the requirements given to the design process, additional, more refined, requirements can be determined. The assumption underlying the refinement of requirements into more specific requirements is that more specific requirements can be used to focus the design process.

**Manipulation of requirements**

On the basis of the given requirements, more refined requirements can be formulated. For the first qualified requirement qr_m1, refinement knowledge is applied which results in the property refinement graphs depicted in Figures 11.2, 11.3, 11.4, 11.5, and 11.6. Requirements on these refined properties are used to construct a design object description.

Figure 12.2 shows the top-level of the property refinement graph for the property related to the qualified requirement qr_m1.

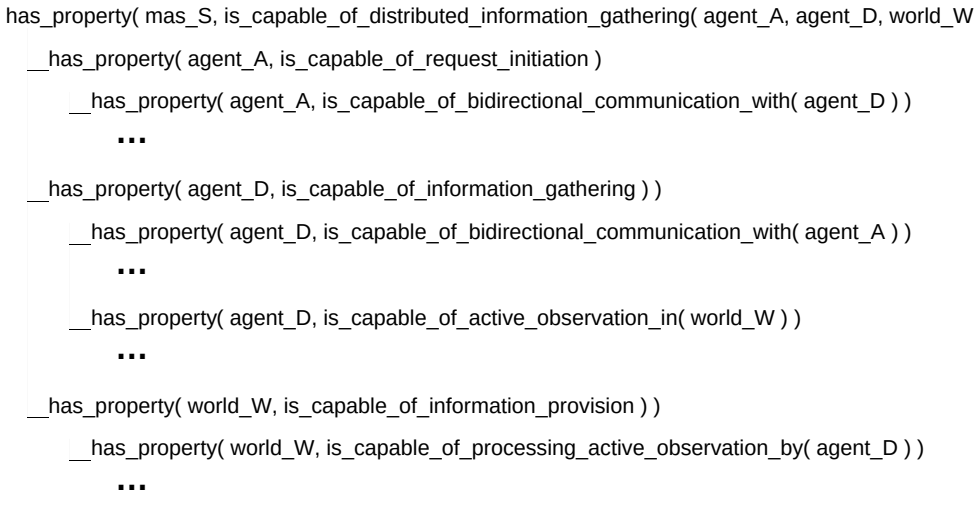


**Figure 12.2** Partial property refinement graph for the property related to qualified requirement qr_m1.

Figure 12.3 shows the refined properties related to the property has_property( agent_A, is_capable_of_bidirectional_communication_with( agent_D ) ).

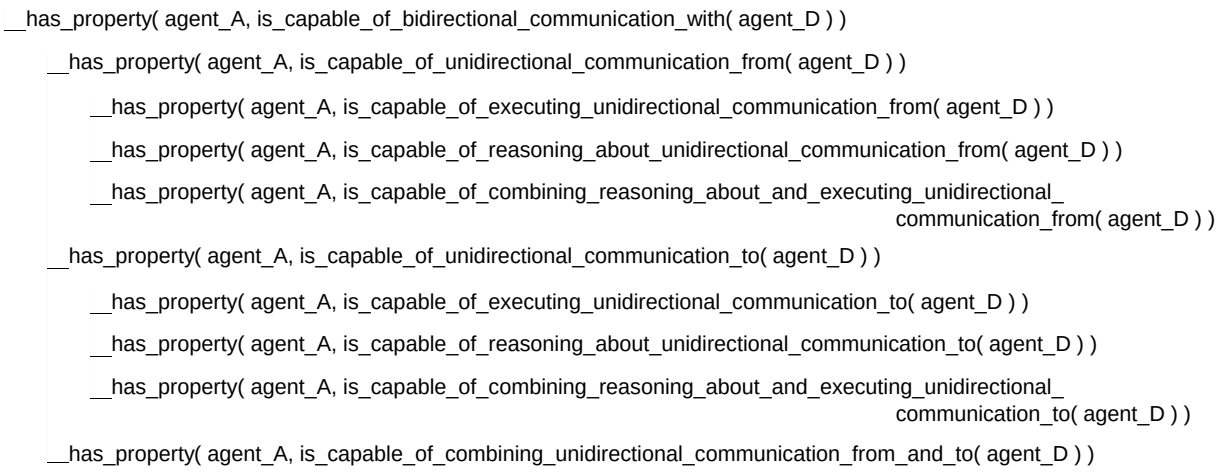


**Figure 12.3** Property refinement graph for has_property( agent_A, is_capable_of_bidirectional_communication_with( agent_D ) ).

Figure 12.4 shows the refined properties related to the property has_property( agent_D, is_capable_of_bidirectional_communication_with( agent_A ) ).

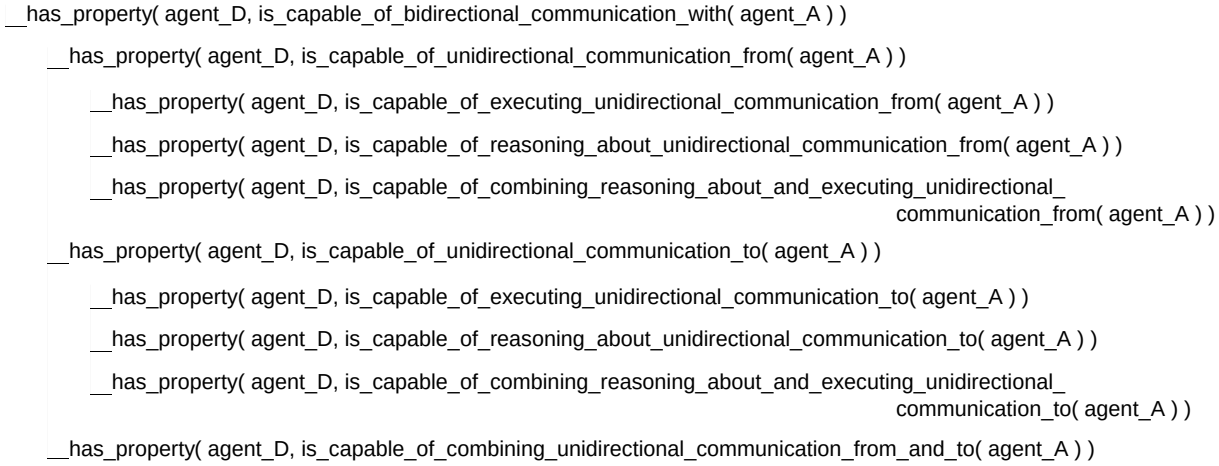


**Figure 12.4** Property refinement graph for has_property( agent_D, is_capable_of_bidirectional_communication_with( agent_A ) ).

Figure 12.5 shows the refined properties related to the property has_property( agent_D, is_capable_of_active_observation_in( world_W ) ).

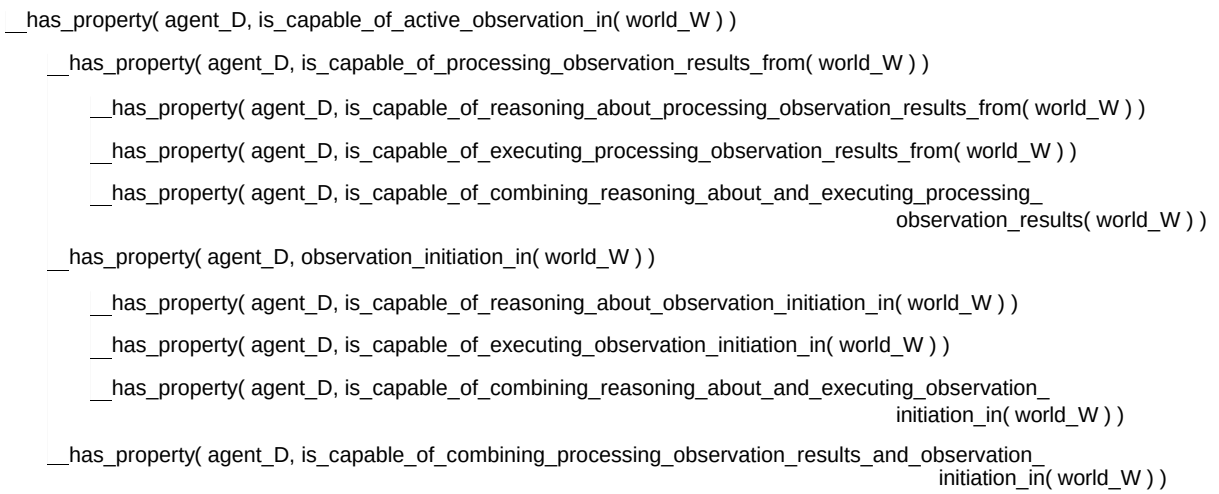


**Figure 12.5** Property refinement graph for has_property( agent_D, is_capable_of_active_observation_in( world_W ) ).

Figure 12.6 shows the refined properties related to the property has_property( world_W, is_capable_of_processing_active_observation_by( agent_D ) ).

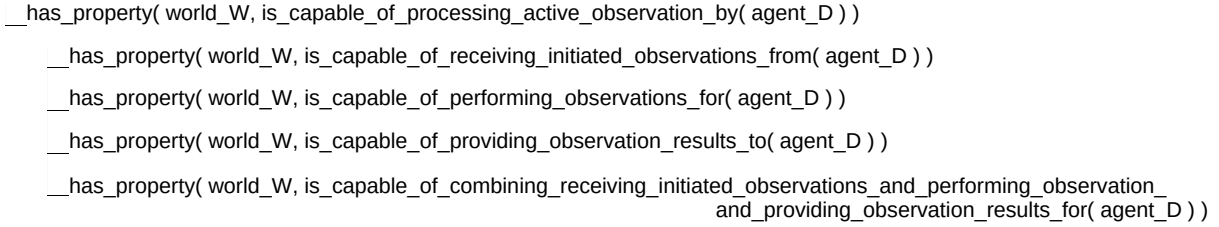


**Figure 12.6** Property refinement graph for has_property( world_W, is_capable_of_processing_active_observation_by( agent_D ) ).

Within the above example the property is capable of active observation in the world is introduced. This property is refined into three specialised properties: the property is capable of observation initiation, the property is capable of processing observation results, and the property is capable of combining process observation results and observation initiation.

Refinement with respect to realisation results in the following refined properties: the property is capable of reasoning about observation initiation in the world, the property is capable of executing observation initiation in the world, the property is capable combining reasoning about and executing observation initiation in the world, the property is capable of reasoning about processing observations results from the world, the property is capable of executing processing observation results from the world, and the property is capable of combining reasoning about and executing processing observation results from the world.

### 12.2.4 Representation of an agent design

The representation of requirements on multi-agent systems has been shown in Section 12.2.2. In this section formal representations of design object descriptions for multi-agent systems are presented. Moreover, knowledge that can be used to derive properties of the design, for example the required properties, is presented.

The implication of designing (parts of) a multi-agent system, is that the multi-agent system itself is the object of design, and as such should be represented in a design object description. In this paper the design object description is assumed to be a compositional object description. The assumption underlying this decision is that a compositional structure facilitates the process of (re-)design.

The description of the compositional system is augmented with a description relating existing structures to generic models. This provides valuable information for the identification of abilities and properties.

**Representation of an agent design**

The design agent needs a representation of a multi-agent system including agents and the external world. To this purpose, a representation based on objects, attributes, and relations is used. Part of the top level of the multi-agent system can be represented as follows:

```
is_top_level(c_00 );
corresponds_with(c_00, mas_S );
corresponds_with(c_01, agent_A );
corresponds_with(c_04, world_W );
has_characterisation( c_00, generic, multi_agent_system );
has_characterisation( c_01, generic, agent );
has_characterisation( c_04, generic, external_world );
corresponds_with( lm_01, active_observations );
has_subcomponent( c_00, c_01 );
has_subcomponent( c_00, c_04 );
has_information_link( c_00, lm_01 );
has_source_component( lm_01, c_01 );
has_destination_component( lm_01, c_04 );
```

Unique identifiers are assigned to components and links so that names of links and components can be reused in several parts of the composition.

### 12.2.5 Manipulation of design object descriptions

The compositional structure of the design object guides the re-design process. Implications of modifications to the compositional structure of a multi-agent system are first explored at the top-level, then one level lower, et cetera.

**Modification of the top-level of the multi-agent system**

The result of modifying the top-level of the multi-agent system is shown in Figure 12.7: on the basis of the initial description of the multi-agent system and refined requirements, a new multi-agent system is proposed which contains agent D.

Note that although agent D has been added and information links are present between D, PA, and EW, the agent D is an empty component at this point in the design process.

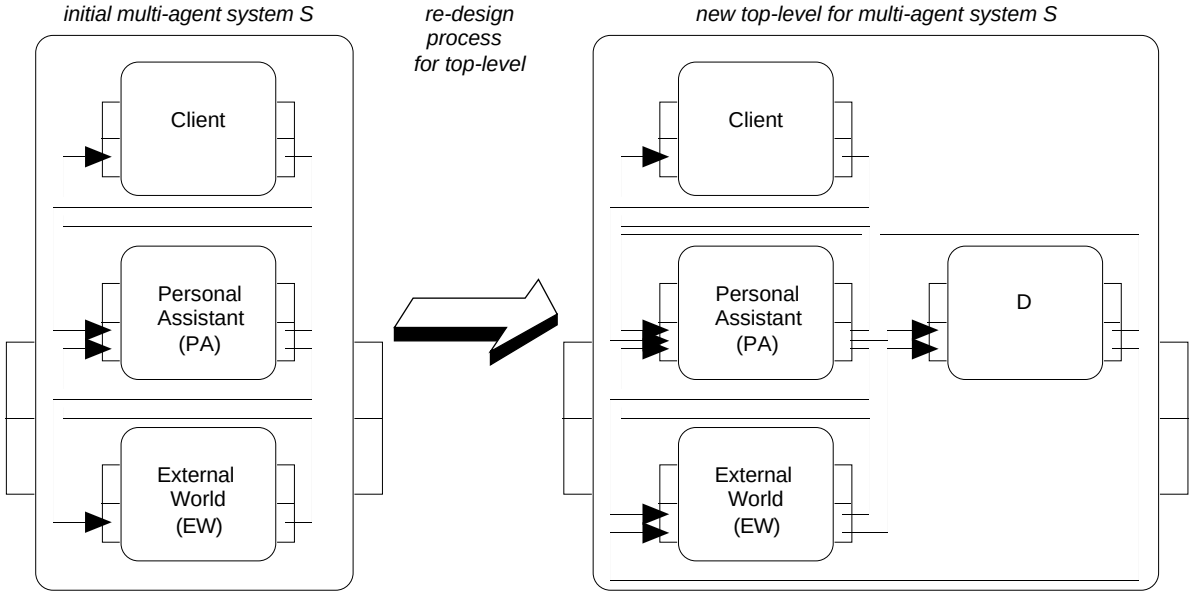


**Figure 12.7** Results of modification to top-level of multi-agent system.

When modifying the description of the agent D, several possible intermediate descriptions are explored during the re-design process. The description of an agent is constructed by modifying previous design object descriptions.

**Modifications within the agent D**

During the re-design process several descriptions of agents are proposed. For example, an agent D (part of design object description no. 14 in Figure 12.8) may be proposed. Structural analysis shows that this particular agent D does have the ability of 'observation initiation', yet lacks the ability of 'bi-directional communication'.

Assessment points, see Section B.2.2, are more specific properties of a design object description, which are related to the current design requirement in focus, yet are simpler to realise ('satisfy') than the design requirement in focus. The design requirement referring to the property is capable of bi-directional communication is related to a number of assessment points, among which: a component is present for bi-directional communication, and private information links are present for bi-directional communication. Non-realised assessment points are not realised at the same time, a strict order on realisations is imposed, so that consequences of realising one assessment point can aid the realisation of another assessment point. Below two knowledge elements are presented which relate assessment points which need to be realised to modifications of the current design object description.

To realise the assessment point a component is present for bi-directional communication for C: component_name, a new component is introduced, which is to be a sub-component of the C: component_name, this component is characterised as being a component for bi-directional communication, and this component is made 'awake' by task control in C: component_name.

```
if   assessment_point_to_be_realised( has_property(
                                        A: component_name,
                                        a_component_is_present_for_bidirectional_communication_
                                        with( A2: component_name ) ) )
     and  current_DOD_contents( has_task_control(
                                        A: component_name,
                                        T: task_control_kb_name ), pos )
then      addition_to_current_DOD( is_component(
                                        new_name( A: component_name, AIM ) ) )
     and  addition_to_current_DOD( has_subcomponent(
                                        A: component_name,
                                        new_name( A: component_name, AIM ) ) )
```

```
and   addition_to_current_DOD( has_characterisation(
                                    new_name( A: component_name, AIM ),
                                    generic,
                                    bidirectional_communication ) )
and   addition_to_current_DOD( has_characterisation(
                                    new_name( A: component_name, AIM ),
                                    specific,
                                    bidirectional_communication_with( A2: component_name ) ) )
and   addition_to_current_DOD( has_part(
                                    T: task_control_kb_name,
                                    makes_awake( new_name( A: component_name, AIM ) ) ) );
```

The second knowledge element requires that the sub-component which is to realise bi-directional communication is already present to formulate information links. To realise the assessment point private information links are present for bi-directional communication with for C: component_name, two information links are introduced, which are private information links of C: component_name. These information links connect A: component_name with the sub-component which is to realise bi-directional communication, and these information links are made 'awake' by task control in C: component_name.

```
if    assessment_point_to_be_realised( has_property(
                                    C: component_name,
                                    private_information_links_are_present_for_bidirectional_
                                    communication_with( C2: component_name ) ) )
and   current_DOD_contents(   is_component(
                                        C: component_name ),
                              pos )
and   current_DOD_contents(   has_interface_information_type(
                                        C: component_name,
                                        input_interface,
                                        Cin: information_type_name ),
                              pos )
and   current_DOD_contents(   has_interface_information_type(
                                        C: component_name,
                                        output_interface,
                                        Cout: information_type_name ),
                              pos )
and   current_DOD_contents(   has_subcomponent(
                                        C: component_name,
                                        D: component_name ),
                              pos )
and   current_DOD_contents(   has_characterisation(
                                        D: component_name,
                                        bidirectional_communication ),
                              pos )
and   current_DOD_contents(   has_characterisation(
                                        D: component_name,
                                        bidirectional_communication_with(
                                                C2: component_name ) ),
                              pos )
and   current_DOD_contents(   has_interface_information_type(
                                        D: component_name,
                                        input_interface,
                                        Din: information_type_name ),
                              pos )
and   current_DOD_contents(   has_interface_information_type(
                                        D: component_name,
                                        output_interface,
                                        Dout: information_type_name ),
                              pos )
and   current_DOD_contents(   has_task_control(
                                        C: component_name,
                                        T: task_control_kb_name ),
                              pos )
```

```
then    addition_to_current_DOD( is_information_link(
                                 new_name( C: component_name, incoming_comm ) ) )
  and   addition_to_current_DOD( has_information_link(
                                 C: component_name,
                                 new_name( C: component_name, incoming_comm ) ) )
  and   addition_to_current_DOD( has_source_component(
                                 new_name( C: component_name, incoming_comm ),
                                 C: component_name ) )
  and   addition_to_current_DOD( has_source_information_type (
                                 new_name( C: component_name, incoming_comm ),
                                 Cin: information_type_name ) )
  and   addition_to_current_DOD( has_destination_component(
                                 new_name( C: component_name, incoming_comm ),
                                 D: component_name ) )
  and   addition_to_current_DOD( has_destination_information_type (
                                 new_name( C: component_name, incoming_comm ),
                                 Din: information_type_name ) )
  and   addition_to_current_DOD( has_part(
                                 T: task_control_kb_name,
                                 makes_awake( new_name( C: component_name,
                                                        incoming_comm ) ) ) )
  and   addition_to_current_DOD( is_information_link(
                                 new_name( C: component_name,  outgoing_comm ) ) )
  and   addition_to_current_DOD( has_information_link(
                                 C: component_name,
                                 new_name( C: component_name,  outgoing_comm ) ) )
  and   addition_to_current_DOD( has_source_component(
                                 new_name( C: component_name,  outgoing_comm ),
                                 D: component_name ) )
  and   addition_to_current_DOD( has_source_information_type (
                                 new_name( C: component_name, incoming_comm ),
                                 Dout: information_type_name ) )
  and   addition_to_current_DOD( has_destination_component(
                                 new_name( C: component_name,  outgoing_comm ),
                                 C: component_name ) )
  and   addition_to_current_DOD( has_destination_information_type (
                                 new_name( C: component_name, incoming_comm ),
                                 Cout: information_type_name ) )
  and   addition_to_current_DOD( has_part(
                                 T: task_control_kb_name,
                                 makes_awake( new_name( C: component_name,
                                                        outgoing_comm ) ) ) );
```

A 'new' agent D (part of design object description no. 23 in Figure 12.8) is shown in which both abilities are incorporated, as required.

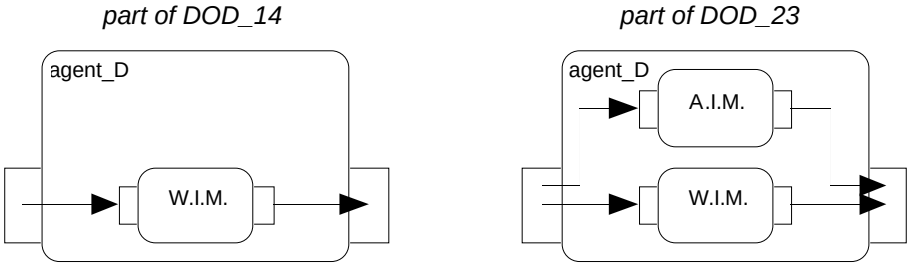


**Figure 12.8** Possible design object descriptions (focussed on composition of agent_D).

Knowledge is employed to analyse any given design object description, to establish whether particular abilities or properties hold. Particular goals, corresponding to the abilities and properties in the current requirements are used to focus this reasoning process.

**Identification of an ability**

As an example of knowledge with which an ability can be identified, consider the follow knowledge elements.

The first knowledge element states that if, in addition to having the necessary task control knowledge to activate the world interaction process and links, the component with identifier C has the generic structure of an agent, includes a component D for world interaction management that is linked to the output interface of the agent via information link Lout, and the agent is linked to the external world via information link Lobs, then the agent C has the ability of executing observation initiation.

```
if  is_component(                          C: component_name )
    and  has_characterisation(             C: component_name,
                                           agent )
    and  has_interface_information_type(   C: component_name,
                                           output_interface,
                                           Cout: information_type_name )
    and  is_component(                     D: component_name )
    and  has_subcomponent(                 C: component_name,
                                           D: component_name )
    and  has_characterisation(             D: component_name,
                                           world_interaction_management )
    and  has_interface_information_type(   D: component_name,
                                           output_interface,
                                           Dout: information_type_name )
    and  is_information_link(              Lout: information_link_name )
    and  has_information_link(             C: component_name,
                                           Lout: information_link_name )
    and  has_source_component(             Lout: information_link_name,
                                           D: component_name )
    and  has_source_information_type(      Lout: information_link_name,
                                           Dout: information_type_name )
    and  has_destination_component(        Lout: component_name,
                                           C: component_name )
    and  has_destination_information_type( Lout: information_link_name,
                                           Cout: information_type_name )
    and  has_task_control(                 C: component_name,
                                           TC: task_control_kb_name )
    and  makes_awake(                      TC: task_control_kb_name,
                                           [ D: component_name, Lout: information_link_name ] )
    and  is_component(                     W: component_name )
    and  has_characterisation(             W: component_name,
                                           external_world )
    and  has_interface_information_type(   W: component_name,
                                           input_interface,
                                           Win: information_type_name )
    and  is_information_link(              Lobs: information_link_name )
    and  has_source_component(             Lobs: information_link_name,
                                           C: component_name )
    and  has_source_information_type(      Lobs: information_link_name,
                                           Cout: information_type_name )
    and  has_destination_component(        Lobs: information_link_name,
                                           W: component_name )
    and  has_destination_information_type( Lobs: information_link_name,
                                           Wout: information_type_name )
then     has_ability(                      C: component_name,
                                           is_capable_of_executing_observation_initiation_in(
                                                 W: component_name ) );
```

The knowledge element below shows how the knowledge on refinement of abilities can also be used to conclude that a more generic ability holds.

```
if   has_ability(                C: component_name,
                                 is_capable_of_reasoning_about_unidirectional_communication_from(
                                       C2: component_name ) )
     and   has_ability(          C: component_name,
                                 is_capable_of_executing_unidirectional_communication_from(
                                       C2: component_name ) )
     and   has_ability(          C: component_name,
                                 is_capable_of_combining_reasoning_about_and_executing_
                                       unidirectional_communication_from( C2: component_name ) )
then      has_ability(           C: component_name,
                                 is_capable_of_unidirectional_communication_from(
                                       C2: component_name ) );
```

When the re-design process has finished, the results include a set of requirements (based on the initial requirements) and a design object description, for example with label dod_55, which fulfils the set of requirements.

The 'size' of the resulting design object description can be 'tuned'. In this situation only the differences between the initial and new multi-agent system are of importance. This includes adding agent D, communication from agent D to agent A and vice versa, interaction from agent A to W and vice versa, plus modifications within agent A and W (to be able to handle agent D).

### 12.2.6 Creation action: realisation of a designed agent

After the design process within the component agent specific task of agent_A has been completed, the agent decides to effectuate the modifications.

As discussed in (Jonker and Treur, 1997) effectuation of the modification of the design can be modelled by changing the material representation of the multi-agent system by performing (material) actions within the external world.

**Changing the material representation of the multi-agent system.**

The resulting design object description, dod_55, contains the difference between the initial multi-agent system and the new multi-agent system (which includes a description of a new agent). The design object properties that together form dod_55, are represented by statements such as:

```
has_DOD_contents( dod_55, is_component( agent_D ), pos );
has_DOD_contents( dod_55, has_subcomponent( agent_C, agent_D ), pos );
```

These statements reside at a meta-level with respect to design object description statements. The second argument of each statement expresses relationships within the design object description.

The design object description of the new multi-agent system is transferred from the agent specific task of agent A to the world interaction management task and to the external world. The modification action itself is selected by the component world interaction management. Together this information specifies the action to be performed.

**Effectuation action for modifying the multi-agent system.**

Within the world interaction management of agent_A, an action is formulated to effectuate the modification of the current multi-agent system:

```
to_be_performed( effectuate_according_to( dod_55 ) );
```

To be able to execute the creation action in the external world, the external world needs to have certain properties. These properties are related to how "equipped" the world is to handle interaction with agents. There are two generic properties needed for the interaction of agents with the external world: the processing of observations and the processing of actions. Observation of the external world was needed to inform agent A of the current material representation of the multi-agent system, see Section 12.2.2.

The property of processing actions can be refined into the properties:

- the external world can receive initiated actions, and the related information, and
- the external world can perform actions (effectuation of the material effects of actions).

To change the number of agents and their characteristics, the external world has to adapt the executable specification of that system while the system is running. This implies that the parts of the system that are affected by the modifications need to be interrupted, their information states stored, after which the executable specification of those parts need to be modified, and the modified system need to be reactivated with the correct information states.

**Result of the effectuation action.**

The external world world_W effectuates the creation action and modifies the multi-agent system according to the given modifications.

As an agent in the multi-agent system, agent D receives a request from agent A (which handles questions from its client agent C): agent A would like to find out more about a particular subject. The agent D gathers information on that subject by initiating observations in the world W, and interpreting the observation results. Once the answer is found, agent D reports its findings to agent A. Agent A can then finally answer the question of the client.

## 12.3 Discussion

Research within multi-agent systems research has focussed on the behaviour of individual agents and their interaction. The dynamic creation of new agents within an existing multi-agent system, on the basis of the identification of newly required functionality and behaviour, is an area on which little research has focussed. Most of the research in the area of dynamic agent creation is based on a genetic programming approach; e.g., (Cetnarowicz, Kisiel-Dorohinicki and Nawarecki, 1996; Numaoka, 1996). The approach taken in this thesis is that to create new agents, an existing agent must be capable of designing a new agent on the basis of a model for design and then be capable of bringing this agent to life.

To design an agent capable of designing another agent, insight is required in the type of agent to be designed. In this thesis a compositional approach to agent design has been followed. An agent's abilities are related to the tasks an agent is able to perform. These abilities are the means with which both the existing agents' abilities and the required abilities of non-existing agents are expressed. In addition, the properties of the multi-agent system and the external world are of importance for the design process.

The architecture of the design agent is based on an existing generic agent model, and includes a refinement of model for re-design of compositional systems, as described in Chapter 10. It combines results from the area of Multi-Agent Systems and the area of AI and Design. The approach described has been formalised: the initial multi-agent system described in this system has been specified and implemented, using the automated implementation generator within the DESIRE software environment, as have the design agent, the new agent and its creation within the new multi-agent system. The formal agent model presented in this chapter includes formalisations of agent design descriptions and requirements on agents represented within an agent, and formalisations of agent design knowledge.

One aspect of the approach described in this chapter is that a design agent not only designs another agent and the implications for the integration of the agent in an existing system at a conceptual level, the design agent also actually creates the new agent dynamically. In fact, a design agent could re-design (parts of) itself in the same manner. The integration of re-design on a conceptual and logical level and run-time modification of the system at the implementation level is an important distinguishing aspect of the approach presented in this paper. This is in contrast to, on the one hand, conceptual and logical approaches for which no direct connection to executable code exists, and, on the other hand, to approaches that address agent creation at an implementation level.

The desiderata dr6 and dr7 have been realised as described by the trace presented in this chapter. The trace illustrates the integration of the result of a re-design process: a design object

(artefact) is realised according to its description. The realisation action modifies the system itself, thereby realising self-modification of the system.

Part **V**

# Discussion & References

In this part the fulfillment of the desiderata distinguished within the overall research theme are discussed with respect to the results presented in this thesis. In Chapter 13 a final discussion is held, suggestions for further research are presented, and conclusions are drawn. The literature referred to in this thesis is listed in the References.

# 13 Discussion, Further Research, and Conclusions

In this chapter the research described in the previous chapters is summarised and the results obtained are related to the central research theme and the more specific research themes.

Section 13.1 summarizes the research presented in this thesis and the results obtained are related to the research themes. In Section 13.2 suggestions for further research are made. In Section 13.3 conclusions are presented, placing the research presented in this thesis in a broader AI perspective.

## 13.1 Discussion

The central research theme, identified in Chapter 2, is:

*How can a compositional structure be used to re-design knowledge-intensive systems?*

The central research theme is demarcated in more detailed desiderata on a process of re-design and knowledge-intensive compositional systems. In addition, a number of desired properties of a design model have been identified, on the basis of which a generic design model has been adopted as the model for design processes in this thesis. This generic design model has been used to construct a model for re-design of compositional systems. In Table 13.1 the desiderata are listed related to chapters of this thesis in which specific desiderata are realised.

| *no.* | *desideratum* | *realised in Chapter* |
|---|---|---|
| **ds1** | unambiguous, formal, representation language | Chapter 4 “Compositional systems: structure” |
| **ds2** | formal semantics | Chapter 4 “Compositional systems: structure” |
| **ds3** | explicit representation of both static and dynamic properties | Chapter 5 “Compositional systems: properties” |
| **ds4** | operationalisation | Chapter 4 “Compositional systems: structure” |
| **ds5** | compositionality as a structuring principle | Chapter 4 “Compositional systems: structure” |
| **dr1** | representation of a knowledge-intensive system as a design object description. | Chapter 7 “A re-design model for compositional systems” |
| **dr2** | representation of qualified requirements on compositional systems | Chapter 7 “A re-design model for compositional systems” |
| **dr3** | model of the manipulation of compositional system structures | Chapter 9 “Design object description manipulation in the re-design model for compositional systems” |
| **dr4** | model of the manipulation of requirements on compositional systems | Chapter 8 “Requirement qualification set manipulation in the re-design model for compositional systems” |
| **dr5** | knowledge on the co-ordination of the re-design process | Chapter 7 “A re-design model for compositional systems” |
| **dr6** | model of integration of design and realisation | Chapter 12 “Re-design of a multi-agent system” |
| **dr7** | model of self-modification | Chapter 12 “Re-design of a multi-agent system” |

**Table 13.1** Relation between desiderata and chapters of this thesis.

In Chapter 3 relevant literature is divided into literature on structuring principles for design processes, and literature on structuring principles for (knowledge-intensive) software systems. A number of essential elements of design processes are distinguished: representation of design objects, representation of qualified requirements, manipulation of design object descriptions, manipulation of sets of (qualified) requirements, and co-ordination of the design process. No

generic design model was found that integrated all of these essential elements of design processes, that was formally described, and that was automatically operationalisable.

Relevant literature on (knowledge-intensive) software systems resulted in a comparison of approaches to structuring principles for knowledge-intensive systems: the modelling frameworks CommonKADS, VITAL, MIKE, and TASK. None of these approaches contain the desired properties expressed in the desiderata.

On the basis of results of the investigation of relevant literature, two important decisions were made:

- the generic model of design introduced in (Brazier, Langen, Ruttkay and Treur, 1994) and described in detail in (Brazier, Langen and Treur, 1999) has been adopted as the model for design processes in this thesis, and
- the DESIRE development method for (knowledge-intensive) compositional systems, of which structuring principles are described in (Brazier, Treur, Wijngaards and Willems, 1995) and temporal semantics are described in (Brazier, Treur, Wijngaards and Willems, 1999) has been adopted as the approach to structuring principles for knowledge-intensive systems in this thesis. In (Brazier, Dunin-Keplicz, Jennings and Treur, 1997) the extension of DESIRE to multi-agent systems is described.

The structure of compositional systems is described in Chapter 4: the structure of compositional systems in the DESIRE development method. Within these compositional systems three design principles are used: process composition, knowledge composition, and the relation between process composition and knowledge composition. In this thesis an unambiguous, formal, representation language is used, with associated formal semantics, which can be operationalised automatically, and has compositionality as a structuring principle. This fulfills desiderata dr1, dr2, dr4, and dr5.

Properties of a compositional system are described in Chapter 5. Properties (and abilities) and relations between properties are described for diagnostic reasoning systems and (parts of) multi-agent systems. These properties are described without reference to DESIRE-specific elements; these properties are reusable across development methods. An explicit representation of these properties, both static and dynamic properties, is described, fulfilling desideratum dr3.

A number of desired properties of a design model are identified in Chapter 2:

- explicit distinction between the manipulation of design object descriptions, manipulation of sets of qualified requirements, and co-ordination of the design process;
- explicit representation and manipulation of design process objectives;
- explicit representation and manipulation of (sets of) qualified requirements; and
- explicit representation and manipulation of design object description.

The generic model of design, described in Chapter 6, realises all of the desired properties. The description of the generic design model in Chapter 6 focusses on the major elements of the generic design model.

The refinement of the generic design model into a model of re-design of compositional systems is described in Chapters 7, 8, and 9. A knowledge-intensive system is represented as a design object description, qualified requirements on compositional systems are explicitly represented, the manipulation of compositional systems and the manipulation of requirements on compositional systems are modelled, and knowledge is identified for the co-ordination of the re-design process: fulfilling desiderata dr1, dr2, dr3, dr4, and dr5.

The progressive refinement steps described in Chapters 7, 8, and 9 are depicted in Figure 13.1. Each of the refinement steps shown in Figure 13.1 is described by a section in Chapter 7, 8, or 9.

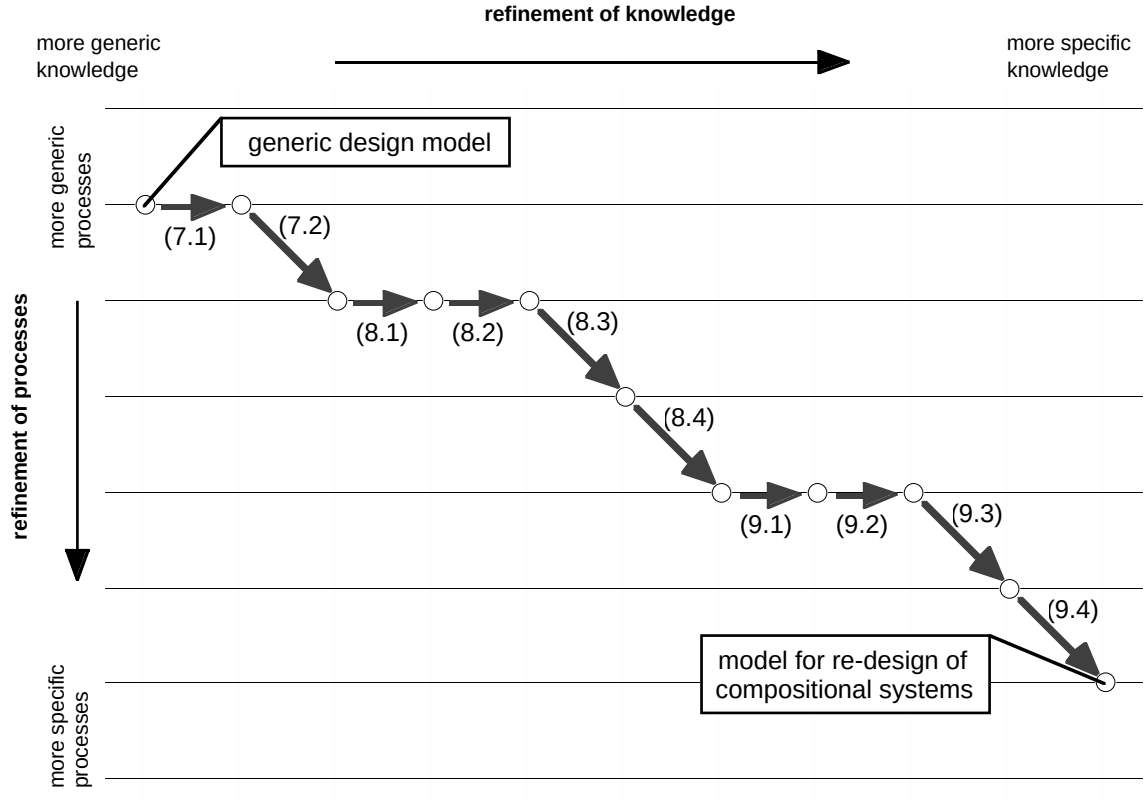


**Figure 13.1** Refinement steps described in Chapters 7, 8, and 9. An arrow indicates a refinement step, its label refers to the section in which the refinement step is described.

Applications of the model of re-design of compositional systems are described in Chapters 9, 10 and 11. The re-design of a diagnostic reasoning system is described in Chapter 11: an existing diagnostic reasoning system is modified on the basis of additional requirements. In Chapter 10, the model for re-design of compositional systems is combined with the generic agent model (described in Section 4.5.2) which yields a design agent specialised in re-design of compositional systems. Chapter 12 describes the application of such a design agent: a self-modifying multi-agent system is the result. The latter application includes a model of integration of design and realisation, and a model of self-modification, which realises the desiderata dr6 and dr7.

The research presented in this thesis is one of three research areas addressed within the REVISE project (Treur, Akkermans, Mars and Wielinga, 1992), a research project on evolutionary design in knowledge-based systems. The three areas of research studied in the REVISE project are: automated re-design of engineering models (see, for example, Pos and Akkermans, 1997), re-design of control structures (see, for example, Straatman, 1997), and re-design of compositional systems. Some other publications on the project are an article on a model including elements of all three re-design models used in the project (Pos, Akkermans and Straatman, 1997) and a comparison of the formal specification languages $(ML)^2$ and DESIRE on the basis of their underlying purposes (REVISE, 1996). The model proposed in (Pos, Akkermans and Straatman, 1997), however, does not realise all of the desired properties of a design model nor does it realise all of the desiderata on a model for re-design of compositional systems.

## 13.2 Further Research

The research presented in this thesis initiated a number of new research questions. Three areas of further research are distinguished: further research in the area of compositional systems, re-design and design processes, and re-design of compositional systems.

Within the area of compositional systems, additional research should address:

- Further identification of generic models and (parts of) specific models which can be employed for the design of compositional systems.
- Further identification of properties and knowledge on properties (e.g., of generic models), which can be employed for the design of compositional systems.
- Integration of different modelling approaches to compositional systems. Within the field of Knowledge Engineering, a number of modelling approaches exist each with their own underlying purposes and realisations of these purposes. Having a means to integrate, (re-)use and exchange models among these modelling approaches would be an important step forward.

Within the area of (re-)design processes additional research should address:

- Distributed design on the basis of a design agent, for example the one described in Chapter 10. The model of the design agent includes explicit reasoning about co-operation (with other agents). This could be employed for distributed design: several design agents co-operate to design a particular object; each design agent has their own speciality (or view). Such an approach could be similar, yet more generic and oriented towards multi-agent systems, than the Single Function Agents approach (Dunskus, Grecu, Brown and Berker, 1995; Berker and Brown, 1996) in which specialised design agents, with a particular view on a design object, negotiate with each other to achieve a design object which satisfies requirements imposed by all design agents.
- Combining research within the field of Requirements Engineering with research on design processes and further extend knowledge on and models for manipulation of requirements.
- The application of the re-design process to other areas, for example, simulation model re-design, re-design of object-oriented software, or areas outside of software processes: machines (e.g., manufacturing), organisations, and buildings to explore differences in design strategies, methods and techniques for manipulation of requirements and design object descriptions, and compositions of the re-design process.

Within the area of the re-design of compositional systems additional research should address:

- Combining research on self-modifying agent systems with machine learning approaches to construct adaptive systems which tune themselves to, e.g., human users,
- Broker agents (e.g., on the Internet) that provide ‘expertise’ to other agents, which modify themselves accordingly,
- Maintenance agents: agents capable of re-designing a specific agent to adapt to new needs,
- Incorporating re-design systems in software development tools (for compositional systems),
- Investigating when and how to co-operate with the user during a process of (re-) design of a compositional system,
- Extending the approach taken in this thesis with a number of other methods and techniques, and investigating the application of these methods and techniques for the manipulation of sets of qualified requirements and the manipulation of design object descriptions.

## 13.3 Conclusions

The research presented in this thesis has provided an answer to the central research theme: how can a compositional structure be used to re-design knowledge-intensive systems? The applications of the model for re-design of compositional systems have illustrated how a compositional structure can be used: compositionality as a structuring principle for describing design processes, and compositionality as a structuring principle for describing design object descriptions.

A prototype implementation of the model for re-design of compositional systems has shown the practical side of the answer to the overall research question. The re-design of a diagnostic reasoning system and a the self-modification of a multi-agent system have been shown to be feasible.

The re-design of compositional systems has been shown to be possible, yet the examples used in this thesis are restricted in the sense that ‘only’ two special cases have been re-designed.

Research within the areas of Artificial Intelligence and Multi-Agent Systems focusses on the identification of properties of compositional systems and results of this research can be used to further augment the applicability of the re-design process as described.

The application of the model of re-design of compositional systems to self-modifying (multi-agent) systems is a step towards flexible, adaptive, systems.

# References


ABEN, M. (1995). *Formal Methods in Knowledge Engineering*. PhD thesis, University of Amsterdam, Faculty of Psychology, February 1995.

AKIN, O. (1978). How do Architects Design. In: LATOMBE, J.C. (Ed.), *Artificial Intelligence and Pattern Recognition in Computer Aided Design*, North Holland, pp. 65-98.

AKKERMANS, H., BORST, P., POS, A., and TOP. J. (1995). Experiences in Conceptual Modelling for a Mechatronic Design Library. In: (Gaines and Musen, 1995), pp. 39/1-39/15.

ALBERTS, L.K., WOGNUM, P.M., and MARS, N.J.I. (1992), Structuring design knowledge on the basis of generic components. In: GERO, J.S. (Ed.), *Artificial Intelligence in Design (AID'92)*, Kluwer, Dordrecht, pp. 639-656.

ALBERTS, L.M., BAKKER, R.R., BEEKMAN, D., and WOGNUM, P.M. (1993). Model-based redesign of technical systems. In: *Proceedings of the 4th international workshop on principles of diagnosis*.

ALBERTS, M. (1993). *YMIR: an ontology for engineering design*. PhD thesis, University of Twente, The Netherlands.

ALLEN, R., and GARLAN, D. (1996). The Wright Architectural Specification Language. *Technical Report CMU-CS-96-TBD*, Carnegie Mellon University.

ALTMEYER, J., and SCHÜRMANN, B. (1996). On design formalisation and retrieval of reuse candidates. In: Gero, J.S., and Sudweeks, F. (Eds.), *Artificial Intelligence in Design, AID'96*, Kluwer Academic Publishers, pp. 231-250.

ANGELE, J., DECKER, S., PERKUHN, R., and STUDER, R. (1996). Modelling Problem-Solving Methods in New KARL. In: GAINES, B.R., and MUSEN, M.A. (Eds.), *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems workshop (KAW'96*), Calgary: SRDG Publications, Department of Computer Science, University of Calgary, pages 1/1-1/18.

ANGELE, J., FENSEL, D., and STUDER, D. (1996). Domain and Task Modelling in MIKE. *Proceedings of the IFIP WG 8.1/13.2 Joint Working Conference, Domain Knowledge for Interactive System Design*. Geneva, Switzerland, May 8-10th.

ASIMOV, I. (1963, 1991). *I Robot*, Bantam Books. Reprint from I Robot, 1963, Garden City, NY, Doubleday.

BAALEN, J. VAN (1992). Automated design of specialised representations. In: *Artificial Intelligence*, **54**, pp. 121-198.

BALL, N.R., and BAUERT, F. (1992). The Integrated Design Framework: supporting the design process using a blackboard system. In: GERO, J.S. (Ed.), *Artificial Intelligence in Design (AID'92)*, Kluwer academic publishers, pp. 327-348.

BARRINGER, H., FISHER, M. , GABBAY, D., OWENS, R., and REYNOLDS, M. (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd., and John Wiley & Sons.

BECK, M., and PARMEE, I. (1997). An Evolutionary Approach to Conceptual Design Exploration. In: CANDY, L., and HORI, K. (Eds.), *Proceedings of the First International Workshop on Strategic Knowledge and Concept Formation*, Lutchi Research Centre, pp. 161-172.

BENJAMINS, V.R. (1993). *Problem Solving Methods for Diagnosis*. PhD Thesis, University of Amsterdam, Amsterdam, The Netherlands.

BENJAMINS, V.R., and ABEN, M. (1997). Structure-preserving knowledge-based system development through reusable libraries: a case study in diagnosis. In: *International journal of human computer studies*, **47**, pp. 259-288.

BENJAMINS, V.R., FENSEL, D., and STRAATMAN, R. (1996). Assumptions of Problem-Solving Methods and their Role in Knowledge Engineering. In: WAHLSTER, W. (Ed.), *Proceedings European Conference on AI (ECAI'96)*, Wiley and Sons, Chichester, pp. 408-412.

BERKER, I., and BROWN, D.C. (1996). Conflicts and Negotiation in Single Function Agent Based Design Systems. In: BROWN, D.C., LANDES, S.E., and PETRIE, C.J. (Eds.), *Concurrent Engineering: Research and Applications, Journal*, Special Issue: Multi Agent Systems in Concurrent Engineering, Technomic Publishing Inc., **4**(1), pp. 17-33.

BERTALANFFY, L. VON (1968). *General Systems Theory: Foundation, Development, Applications*. G. Braziller, New York.

BEYS, P., BENJAMINS, V.R., and VAN HEIJST, G. (1996). Remedying the reusabiltiy - usability tradeoff for problem-solving methods. In: GAINES, B.R., and MUSEN, M.A. (Eds.), *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems workshop (KAW'96*), Calgary: SRDG Publications, Department of Computer Science, University of Calgary, pp. 2/1-2/20.

BIGGERSTAFF, T.J. (1992). An Assessment and Analysis of Software Reuse. In: YOVITS, M.C. (Ed.), *Advances in Computers*, **34**, Academic Press inc, pp. 1-57.

BLAMEY, S. (1986). Partial Logic. In: GABBAY, D., and GÜNTHNER, F. (Eds.), *Handbook of Philosophical Logic.* Dordrect, **III**, pp. 1-70, Reidel, Dordrecht.

BOELENS, J. (1991). *Soil sanitation and strategic interaction*. Masters thesis, department of mathematics and computer science, vrije universiteit amsterdam, 1991.

BOOCH, G. (1991). *Object oriented design with applications*. Benjamins Cummins Publishing Company, Redwood City.

BORST, W.N., AKKERMANS, J.M., and TOP, J.L. (1997). Engineering ontologies. In: *International Journal of Human-Computer Studies*, special issue on using explicit ontologies in KBS development, **46**, pp. 365-406.

BOY, G.A. (1995). Agent-oriented interaction in cooperative computer-based training systems. In: (Gaines and Musen, 1995), pp. 19/1-19/10.

BRADSHAW, J.M. (1997) (Ed.). *Software Agents*. AAAI Press / MIT Press.

BRAZIER, F.M.T., and WIJNGAARDS, N.J.E. (1997). An instrument for a purpose driven comparison of modelling frameworks. In. PLAZA, E., and BENJAMINS, R. (Eds). *Proceedings of the 10th European Workshop on Knowledge Acquisition, Modelling, and Management (EKAW'97)*. Sant Feliu de Guixols, Catalania, Lecture Notes in Artificial Intelligence, **1319**, Springer Verlag, pp. 323-328.

BRAZIER, F.M.T., and WIJNGAARDS, N.J.E. (1998). A Purpose Driven Method for the Comparison of Modelling Frameworks. In: (Gaines and Musen, 1998), pp. 18.

BRAZIER, F.M.T., CORNELISSEN, F., GUSTAVSSON, R., JONKER, C.M., LINDEBERG, O., POLAK, B., and TREUR, J. (1998). Agents Negotiating for Load Balancing of Electricity Use. In: PAPAZOGLOU, M.P. (Ed.), *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, IEEE Computer Society Press, pp. 622-629.

BRAZIER, F.M.T., DUNIN-KEPLICZ, B.M., JENNINGS, N.R., and TREUR, J. (1997). Formal specification of multi-agent systems: a real world case. in: LESSER, V. (Ed.), *Proceedings of the first international conferences on multi-agent systems (ICMAS'95)*, MIT Press, pp. 25-32. Extended version in: HUHNS, M., and SINGH, M. (Eds.), *International Journal of Co-operative Information Systems (IJCIS)*, **6**(1), (1997), Special Issue on Formal Methods in Co-operative Information Systems: Multi-Agent Systems, pp. 67-94.

BRAZIER, F.M.T., JONKER, C.M., and TREUR, J. (1996). Formalisation of a co-operation model based on joint intentions. In: MÜLLER, J.-P., WOOLDRIDGE, M., and JENNINGS, N.R. (Eds.), *Intelligent Agents III, Proceedings of the third international workshop on Agent Theories, Architectures and Languages, (ATAL'96)*. Lecture Notes in Artificial Intelligence, **1193**, Springer-Verlag, pp. 141-155.

BRAZIER, F.M.T., JONKER, C.M., and TREUR, J. (1998). Principles of Compositional Multi-agent System Development. In: CUENA, J. (Ed), *Proceedings of the 15th IFIP World Computer Congress WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98*, Chapman and Hall, pp. 347-360.

BRAZIER, F.M.T., JONKER, C.M., TREUR, J., and WIJNGAARDS, N.J.E. (1998a). An Agent Architecture for Dynamic Re-design of Agents. In: RODGERS, P., and HUXOR, A. (Eds.), *Proceedings of the AID'98 Workshop on Distributed Web-based Design Tools*, Lisbon. Extended abstract in: DEMAZEAU, Y. (Ed.), *Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS'98*, IEEE Computer Society Press, pp. 401-402.

BRAZIER, F.M.T., JONKER, C.M., TREUR, J., and WIJNGAARDS, N.J.E. (1998b). On the role of abilities of agents in redesign. In: (Gaines and Musen, 1998), pp. 16.

BRAZIER, F.M.T., JONKER, C.M., TREUR, J., and WIJNGAARDS, N.J.E. (1998c). Compositional Design of a Generic Design Agent. In: LUGER, G. and INTERRANTE, L. (Eds.), *Proceedings of the AAAI Workshop on Artificial Intelligence and Manufacturing: State of the Art and Practice*, AAAI Press, pp. 30-39.

BRAZIER, F.M.T., JONKER, C.M., TREUR, J., and WIJNGAARDS, N.J.E. (1999a). Deliberate Evolution in Multi-Agent Systems. In: *Proceedings of the third International Conference on Autonomous Agents, AGENTS'99*. ACM Press, to appear.

BRAZIER, F.M.T., JONKER, C.M., TREUR, J., and WIJNGAARDS, N.J.E. (1999b). On the Use of Shared Task Models in Knowledge Acquisition, Strategic User Interaction and Clarification Agents. *International Journal of Human-Computer Studies*. In press. Specific versions on shared task models and strategic user interaction appeared in (Brazier, Treur and Wijngaards, 1996*a*) and (Brazier, Treur and Wijngaards, 1996*b*).

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, and TREUR, J. (1996). Logical theory of design. In: (Gero, 1996), pp. 243-266.

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, and TREUR, J. (1997a). Modelling conflict management in design: an explicit approach. In: SMITH, I.F.C. (Ed.), *AIEDAM*, Special issue on conflict management in design, **9**(4), pp. 355-366.

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, and TREUR, J. (1997b). A compositional approach to modelling design rationale. In: CHUNG, P.W.H., and BANARES-ALCANTARA, R., (Eds.), *Artificial Intelligence for Engineering Design, Analysis and Manifacturing (AIEDAM)*, Special Issue on Representing and Using Design Rationale, **11**(2), pp. 125-139.

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, and TREUR, J. (1998). Strategic Knowledge in Compositional Design Models. In: GERO, J.S., and SUDWEEKS, F. (Eds.), *Proceedings of the Fifth International Conference on Artificial Intelligence in Design (AID'98)*, Kluwer Academic Publishers, Dordrecht, in press, pp. 18.

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, and TREUR, J. (1999). Generic model of design. Technical report. Department of Artificial Intelligence, Faculty of Sciences, Vrije Universiteit Amsterdam, The Netherlands.

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, RUTTKAY ZS., and TREUR, J. (1994). On formal specification of design tasks. In GERO, J.S., and SUDWEEKS, F. (Eds.), *Artificial Intelligence in Design (AID'94)*, Kluwer Academic Publishers, Dordrecht, pp. 535-552.

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, TREUR, J., and WIJNGAARDS, N.J.E. (1996). Redesign and reuse in compositional knowledge-based systems. In WOGNUM, P.M., and SMITH, I.F.C. (Eds.), *Knowledge Based Systems*, Special issue on models and techniques for reuse of designs, **9**, pp. 105-118.

BRAZIER, F.M.T., LANGEN, P.H.G. VAN, TREUR, J., WIJNGAARDS, N.J.E., and WILLEMS, M. (1996). Modelling an elevator design task in DESIRE: the VT example. In: (Schreiber and Birmingham, 1996), pp. 469-520.

BRAZIER, F.M.T., TREUR, J. (1994). User centered knowledge-based system design: a formal modelling approach. In: STEELS, L., SCHREIBER, G., and VAN DE VELDE, W. (Eds.), *Proceedings of the 8th European Knowledge Acquisition Workshop*, Lecture Notes in Artificial Intelligence, **867**, Springer-Verlag, pp. 283-302.

BRAZIER, F.M.T., TREUR, J., and WIJNGAARDS, N.J.E. (1996a). The Elicitation of a Shared Task Model. In: SHADBOLT, N., O'HARA, K., and SCHREIBER, A.TH. (Eds), *Advances in Knowledge Acquisition. 9th European Knowledge Acquisition Workshop, EKAW'96*. Lecture Notes in Artificial Intelligence, **1076**, pp. 278-289.

BRAZIER, F.M.T., TREUR, J., and WIJNGAARDS, N.J.E. (1996b). Interaction with experts: the role of a shared task model. In: WAHLSTER, W. (Ed.), *Proceedings European Conference on AI (ECAI'96)*, Wiley and Sons, Chichester, pp. 241–245.

BRAZIER, F.M.T., TREUR, J., WIJNGAARDS, N.J.E., and WILLEMS, M. (1995). Formal specification of hierarchically (de)composed tasks. In (Gaines and Musen, 1995), pp. 25/1-25/20.

BRAZIER, F.M.T., TREUR, J., WIJNGAARDS, N.J.E., and WILLEMS, M. (1999). Temporal semantics of compositional task models and problem solving methods. In: *Data and Knowledge Engineering*, **29**, pp. 17-42.

BREUKER, J. (1994). Components of Problem Solving and Types of Problems. In: STEELS, L., SCHREIBER, G., and VAN DE VELDE, W. (Eds.), *Proceedings of the 8th European Knowledge Acquisition Workshop*, Lecture Notes in Artificial Intelligence, **867**, Springer-Verlag, pp. 118-136.

BREUKER, J.A. (1997). Problems in indexing problem solving methods. In: FENSEL, D. (Ed.), *Proceedings of the Problem Solving Methods for Knowledge Based Systems workshop (IJCAI'97)*, pp. 19-35.

BROWN, D., and CHANDRASEKARAN, B. (1989). *Design problem solving: knowledge structures and control strategies*, San Mateo, CA, Morgan Kaufmann.

BROWN, D.C., and BIRMINGHAM, W.P. (1997). Understanding the Nature of Design. In: *IEEE Expert*, **12**(2), pp. 14-16.

BRUMSEN, H.A., PANNEKEET, J.H.M., and TREUR, J. (1992). A compositional knowledge-based architecture modelling process aspects of design tasks, *Proceedings of the 12th International Conference on Artificial Intelligence, Expert systems and Natural Language (Avignon'92)*, EC2, Nanterre, **1**, pp. 283-293.

CARBONELL, J.G. (1983). Derivational Analogy and its Role in Problem Solving. In: *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI'83)*, Morgan Kaufmann Publisers, Inc., pp. 64-69.

CETNAROWICZ, K., KISIEL-DOROHINICKI, M., and NAWARECKI, E. (1996). The Application of Evolution Process in Multi-Agent World to the Prediction System. In: TOKORO, M., (Ed.), *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS'96)*, MIT/AAAI Press, Menlo Park CA, pp. 26-32.

CHANDRASEKARAN, B. (1986). Generic Tasks in Knowledge-Based Reasoning: High Level Building Blocks for Expert System Design. In: *IEEE Expert*, pp. 23-30.

CHANDRASEKARAN, B. (1990). Design problem solving: a task analysis. In: *AI Magazine*, **11**(4), 59-71.

COMPTON, P., RAMADAN, Z., PRESTON, P., LE-GIA, T., CHELLEN, V., MULHOLLAND, M., HIBBERT, D.B., HADDAD, P.R., and KANG, B. (1998). A trade-off between domain knowledge and problem-solving method power. In: (Gaines and Musen, 1998), pp. 19.

CONSOLE, L., and TORASSO, P. (1990). Hypothetical reasoning in causal models. In: *International Journal of Intelligent Systems*, **5**(1), pp. 83-124.

CORNELISSEN, F., JONKER, C.M., and TREUR, J. (1997). Compositional Verification of Knowlegde-Based Systems: A Case-Study for Diagnostic Reasoning. In: PLAZA, E., and BENJAMINS, R. (Eds.), *Knowledge Acquisition, Modeling and Management*, proceedings of the 10th European workshop, EKAW'97, Lecture Notes in Artificial Intelligence, **1319**, Springer, Berlin, pp. 65-80.

COYNE, R.D., ROSENMAN, M.A., RADFORD, A.D., BALACHANDRAN M., and GERO, J.S. (1990), *Knowledge-based design systems*, Addison-Wesley Publishing Company, Reading.

DAUBE, F., and HAYES-ROTH, B. (1989). A Case-Based Mechanical Redesign System. In: *Proceedings of the 11th IJCAI*, pp. 1402-1407.

DAVIS, A.M. (1993). *Software Requirements: objects, functions, and states*. Prentice Hall PTR, New Jersey.

DEARDEN, A.M., and HARRISON, M.D. (1993). Using previous experience in similarity assessment for CBR: A formal treatment. In: *Proceedings of the IJCAI'93*.

DENNETT, D.C. (1987). *The Intentional Stance*. MIT Press, Cambridge.

DIXON, J.R. (1989), On research methodology towards a scientific theory of engineering design. In NEWSOME, S.L., SPILLERS, W.R., and FINGERS, S. (Eds.) *Design Theory '88* , Springer Verlag, N.Y., pp. 316-337.

DUFFY, A.H.B. (1997). The "What" and "How" of Learning in Design. In: *IEEE Expert*, **12**(3), pp. 71-76.

DUNSKUS, B.V., GRECU, D.L., BROWN, D.C., and BERKER, I. (1995). Using Single Function Agents to Investigate Conflict. In: *Artificial Intelligence in Engineering Design and Manufacturing (AIEDAM)*, Special Issue: Conflict Management in Design, **9**(4), pp. 299-312.

EDMONDS, E.A., CANDY, L., JONES, R., and SOUFI, B. (1994). Support for Collaborative Design: Agents and Emergence. In: *Communications of the ACM*, **37**(7), pp. 41-47.

ELST, J. VAN DEN, HARMELEN, F. VAN, SCHREIBER, G., and THONNAT, M. (1995). A functional specification of reusing software components. In: *Proceedings of the sixth international conference on software engineering and knowledge engineering*, Knowledge Science Institute, pp. 374-381.

ENGELFRIET J., and TREUR J. (1994). Temporal Theories of Reasoning, In: MACNISH, C., PEARCE, D., PEREIRA, L.M. (Eds.), *Logics in Artifical Intelligence, Proceedings of the 4th European Workshop on Logics in Artificial Intelligence (JELIA'94)*, Lecture Notes in AI, **838**, Springer Verlag, pp. 279-299.

ERIKSSON, H., PUERTA, A.R., GENNARI, J.H., ROTHENFLUH, T.E., TU, S.W., and MUSEN, M.A. (1995). Custom-Tailored Development Tools for Knowledge-Based Systems. In: (Gaines and Musen, 1995), pp. 26/1-26/19.

FALKENHAINER, B., and FORBUS, K.D. (1991). Compositional modeling: finding the right model for the job. In: *Artificial Intelligence*, **51**(1-3), pp. 95-143.

FALKENHAINER, B., and FORBUS, K.D. (1992). Composing task-specific models. In: *Automated Modelling*, DSV **41**, ASME, pp. 1-9.

FENSEL, D. (1995). *The Knowledge Acquisition and Representation Language KARL*. Kluwer Academic Publisher, Boston, 1995.

FENSEL, D. (1997). The Tower-of-Adapters Method for Developing and Reusing Problem-Solving Methods. In: PLAZA, E., and BENJAMINS, R. (Eds.), *Knowledge Acquisition, Modeling and Management, proceedings of the 10th European workshop (EKAW'97)*, Lecture Notes in Artificial Intelligence, **1319**, Springer, Berlin, pp. 97-112.

FENSEL, D., and HARMELEN, F. VAN (1994). A comparison of languages which operationalise and formalise KADS models of expertise. In: *The Knowledge Engineering Review*, **9**, pp. 105-146.

FENSEL, D., and MOTTA, E. (1998). Structured development of problem solving methods. In: (Gaines and Musen, 1998), pp. 20.

FENSEL, D., and STRAATMAN, R. (1998). The Essence of Problem-Solving Methods: Making Assumptions to Gain Efficiency. In: *International Journal of Human-Computer Studies (IJHCS)*, **48**(2), pp. 181-215.

FENSEL, D., MOTTA, E., DECKER, S., and ZDRAHAL, Z. (1997). Using Ontologies for Defining Tasks, Problem-Solving Methods and their Mappings. In: PLAZA, E., and BENJAMINS, R. (Eds.), *Knowledge Acquisition, Modeling and Management, proceedings of the 10th European workshop (EKAW'97)*, Lecture Notes in Artificial Intelligence, **1319**, Springer, Berlin, pp. 113-128.

FISCHER, G., and LEMKE, A.C. (1988). Construction kits and design environments: steps toward human problem-domain communication. In: *Human-Computer Interaction*, **3**, pp. 179-222.

FISCHER, G., and STEVENS, C. (1991). Information Access in Complex, Poorly Structured Information Spaces. In: *Communications of the ACM*, **4**, pp. 63-70.

FISCHER, G., HENNINGER, S., and REDMILES, D. (1991). Intertwining Query Construction and Relevance Evaluation. In: *Communications of the ACM*, pp. 55-62.

FISHER, M. (1993). Concurrent MetateM - a language for modelling reactive systems. In: *Parallel architectures and languages in Europe (PARLE)*, Munich, Germany, Springer Verlag, Lecture Notes in Computer Science, **694**.

FISHER, M. (1994). A survey of Concurrent MetateM - the language and its applications. In: *Proceedings of the first international conference on temporal logic*, Springer Verlag, Lecture Notes in Computer Science, **827**.

FORBUS, K.D. (1988). Intelligent Computer Aided Engineering. In: *AI Magazine*, pp. 23-36.

FORD, K.M., BRADSHAW, J.M., ADAMS-WEBBER, J.R., and AGNEW, N.M. (1993). Knowledge acquisition as a constructive modeling activity. In FORD, K.M., and BRADSHAW, J.M. (Eds), *Int. J. Intelligent Systems*, Special Issue on Knowledge Acquisition as Modeling, **8**(1), pp. 9–23.

FRENCH, R., and MOSTOW, J. (1985). Toward Better Models of the Design Process. In: *AI Magazine*, **6**(1), pp. 44-57.

GAINES, B.R., and MUSEN, M.A. (1995) (Eds.), *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-based Systems workshop (KAW'95*), Calgary: SRDG Publications, Department of Computer Science, University of Calgary.

GAINES, B.R., MUSEN, M.A. (1998) (Eds.). *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-based Systems workshop (KAW'98)* , Calgary: SRDG Publications, Department of Computer Science, University of Calgary.

GALSEY, A., SCHWABACHER, M., and SMITH, D. (1996). Using Modelling Knowledge to Guide Design Space Search. In: GERO, J.S., and SUDWEEKS, F. (Eds.), *Artificial Intelligence in Design (AID'96)*, Kluwer Academic Publishers, pp. 367-385.

GAMMA, E., HELM, R., JOHNSON, R., and VLISSIDES, J. (1994). *Design Patterns: Elements of reusable object-oriented software*. Addison Wesley Longman, Reading, Massachusetts.

GARLAN, D., ALLEN, R., and OCKERBLOOM, J. (1994). Exploiting Style in Architectural Design Environments. In: *Proceedings of the ACM SIGSOFT'94 Symposium on Foundations of Software Engineering*, New Orleans, LA, pp. 14.

GARLAN, D., ALLEN, R., and OCKERBLOOM, J. (1995). Architectural Mismatch or Why it's hard to build systems out of existing parts. In: *Proceedings of the seventh international conference on software engineering*, Seatlle WA, pp. 179-185.

GARLAN, D., and SHAW, M. (1994). An Introduction to Software Architecture. *Technical Report CMU-CS-94-166*, Carnegie Mellon University.

GAVRILA I.S., and TREUR J. (1994). A formal model for the dynamics of compositional reasoning systems. In: COHN, A.G. (Ed.), *Proc. 11th European Conference on Artificial Intelligence (ECAI'94)*, Wiley and Sons, pp. 307-311.

GEBHARDT, F. (1997). Survey on structure-based case retrieval. In: *The Knowledge Engineering Review*, **12**(1), pp. 41-58.

GEELEN, P.A., and KOWLCZYK, W. (1992). A knowledge-based system for the routing of international blanc payment orders. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence, Expert Systems and Natural Language (Avignon'92)*, Nanterre: EC2, **1**, pp. 669-677.

GEELEN, P.A., RUTTKAY, Zs., and TREUR, J. (1991). Logical Analysis and specification of an office assignment task. *Technical report IR-283*, Amsterdam: Vrije Universiteit, Department of Mathematics and Computer Science, AI Group.

GENNARI, J.H., ALTMAN, R.B., and MUSEN, M.A. (1994). Reuse with PROTÉGÉ-II: From Elevators to Ribosomes. *Knowledge systems laboratory, KSL-94-71*, Stanford University School of Medicine.

GERO, J.S. (1990). Design prototypes: a knowledge representation schema for design. In: *AI Magazine*, **11**(4), 26-36.

GERO, J.S. (Ed.) (1996). *Advances in formal design methods for CAD*. Chapman and Hall, London, 299 pp.

GIL, Y., and TALLIS, M. (1995). Transaction-Based Knowledge Acquisition: Complex Modifications Made Easier. In: (Gaines and Musen, 1995), pp. 28/1-29/18.

GOEL, A., and CHANDRASEKARAN, C. (1989). Functional representation of design and redesign problem solving. In: *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, Detroit MI, Morgan Kaufmann Publishers, Los Altos CA, pp. 1388-1394.

GRUBER, T.R. (1990). Acquiring strategic knowledge from experts. In: BOOSE, J.H., and GAINES, B.R. (Eds.), *The Foundations of Knowledge Acquisition*, Knowledge Based Systems, **4**, Academic Press Limited, pp. 115-133.

GRUBER, T.R. (1993). A translation approach to portable ontology specifications. In: *Knowledge Acquisition*, **5**(2), pp. 199-220.

GRUBER, T.R., and OLSEN, G.R. (1994). An Ontology for Engineering Mathematics. In: DOYLE, J., TORASSO, P., and SANEWALL, E. (Eds.), *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann, pp. 241–245.

HARMELEN, F. VAN, and BALDER, J. (1992). $(ML)^2$: A formal language for KADS models of expertise. In: *Knowledge Acquisition*, **4**, pp. 127-161.

HARMELEN, F. VAN, and FENSEL, D. (1995). Formal methods in knowledge engineering. In: The Knowledge Engineering Review, **10**(4), pp. 345-360.

HARMELEN, F. VAN, and TEIJE, A. TEN (1998). Characterising Problem Solving Methods by gradual requirements: overcoming the yes/no distinction. In: (Gaines and Musen, 1998), pp. 15.

HARMELEN, F. VAN, WIELINGA, B., BREDEWEG, B., SCHREIBER, G., KARBACH, W., REINDERS, M., VOß, A., AKKERMANS, H., Bartsch-Spörl, B., and Vinkhuyzen, E. (1992). Knowledge-level reflection. In: LE PAPE, B., and STEELS, L. (Eds.), *Enhancing the knowledge engineering process - contributions from ESPRIT*, Elsevier Science, Amsterdam, pp. 175-204.

HÄUSLEIN, A., and PAGE, B. (1991). Knowledge-Based Approaches to Modelling and Simulation Support. In: *Systematic Analysis of Modeling Simulation*, **8**(4/5), pp. 257-272.

HEIJST, G. VAN, TERPSTRA, P., WIELINGA, B., and SHADBOLT, B. (1992). Using generalised directive models in knowledge acquisition. In: WETTER, TH., ALTHOFF, K.D., BOOSE, J., GAINES, B.G., LINSTER, M., and SCHMALHOFER, F. (Eds.), *Current Developments in Knowledge Acquisition: EKAW'92*, Berlin/Heidelberg, Springer Verlag, pp. 112-132.

HEITMEYER, C.L., JEFFORDS, R.D., and LABAW, B.G. (1996). Automated Consistency Checking of Requirements Specifications. In: *ACM Transactions on Software Engineering and Methodology*, **5**(3), pp. 231-261.

HOOG, R. DE, MARTIL, R., WIELINGA, B.J., TAYLOR, R., BRIGHT, C., and VELDE, W. VAN DE (1994). The CommonKADS model set. *Technical Report KADS-II/M1/DM1.1b/UvA/018/6.0/FINAL*, SWI, University of Amsterdam.

HORI, K. (1997). Where is, What is, and How can we use Strategic Knowledge?. In: CANDY, L., and HORI, K. (Eds.), *Proceedings of the First International Workshop on Strategic Knowledge and Concept Formation*, Lutchi Research Centre, pp. 35-42.

HUNT, J., and MILES, R. (1994). Hybrid case-based reasoning. In: *The Knowledge Engineering Review*, **9**(4), pp. 383-397.

JACKSON, M.A. (1975). *Principles of Program Design*, Academic Press.

JENNINGS, N.R., and WOOLDRIDGE, M.J. (1998) (Eds.). *Agent Technology; Foundations, Application, and Markets*. Springer Verlag.

JONKER, C.M., and TREUR, J. (1997). Modelling an Agent's Mind and Matter. In: BOMAN, M., VELDE, W. VAN DE, (Eds.), *Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'97*, Lecture Notes in AI, **1237**, Springer Verlag, pp. 210-233.

JONKER, C.M., and TREUR, J. (1998a). A Generic Architecture for Broker Agents. In: NWANAM H.S., and NDUMU, D.T. (Eds.), *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'98)*, The Practical Application Company Ltd, pp. 623-624.

JONKER, C.M., and TREUR, J. (1998b). Compositional design and maintenance of broker agents. In: CUENA, J. (Ed.), *Proceedings of the 15th IFIP World Compter Congress, WCC'98 Conference on Information Technology and Knowledge Systems (IT&KNOWS'98)*, Chapman and Hall, to appear, pp. 361-374.

JONKER, C.M. and TREUR, J. (1998c). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: ROEVER, W.P. DE, LANGMAACK, H., and PRUELI, A. (Eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97, Springer Verlag. Lecture Notes in Computer Science, **1536**, pp. 350-380.

JONKER, C.M., and TREUR, J. (1999). Generic model of Diagnosis. Technical Report. Department of Artificial Intelligence, Faculty of Sciences, Vrije Universiteit Amsterdam, The Netherlands.

JONKER, C.M., KREMER, R., LEEUWEN, P. VAN, PAN D., and TREUR, J. (1998). Mapping Visual to Textual Representation of Knowledge in DESIRE. In: (Gaines and Musen, 1998), pp. 20.

KANG, B.H., COMPTON, P., and PRESTON, P. (1998). Simulated Expert Evaluation of Multiple Classification Ripple Down Rules. In: (Gaines and Musen, 1998), pp. 19.

KAUTZ, H.A., SELMAN, B., and COEN, M. (1994). Bottom-Up Design of Software Agents. In: *Communications of the ACM*, **37**(7), pp. 143-146.

KLINKER, G., GENETET, S., and MCDERMOTT, J. (1990). Knowledge Acquisition for evaluation systems. In: BOOSE, J.H., GAINES, B.R. (Eds.), *The foundations of knowledge acquisition, knowledge based systems*, Academic Press Limited, **4**, pp. 269-285.

KNOBLOCK, C.A., and AMBITE, J.L. (1997). Agents for Information Gathering. In: (Bradshaw, 1997), pp. 347-373.

KOLLER, R. (1985). *Für den Machinenbau*. Springer-Verlag, Berlin.

KOLODNER, J. (1993). *Case-based Reasoning*. Morgan Kaufmann Publishers, San Mateo, Ca.

KORF, R.E. (1980). Toward a Model of Representation Changes. In: *Artificial Intelligence*, **14**, pp. 41-78.

KOWALCZYK, W., and TREUR, J. (1990). On the use of a formalised generic task model in knowledge acquisition. In: WIELINGA, B.J., BOOSE, J., GAINES, B., SCHREIBER, G., and VAN SOMEREN, M. (Eds.), *Proceedings of the 4th European Knowledge Acquisition Workshop EKAW'90*, IOS Press Amsterdam, pp. 198-221.

KRUEGER, C.W. (1992). Software Reuse. In: *ACM Computing Surveys*, **24**(2), pp. 131-183.

KUMAR, A.N. (1994). Function based reasoning. In: *The Knowledge Engineering Review*, **9**(3), pp. 301-304.

LANDER, S.E. (1997). Issues in Multi-agent Design Systems. In: *IEEE Expert*, **12**(2), pp. 18-26.

LANGEN, P.H.G. VAN, and J. TREUR (1989). Representing World Situations and Information States by Many-Sorted Partial Models. *Technical Report PE8904*, University of Amsterdam, Department of Mathematics and Computer Science.

LANGEVELDE, I.A. VAN, PHILIPSEN, A.W., and TREUR, J. (1992). Formal specification of compositional architectures. In NEUMANN, B. (Ed.), *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92)*, Wiley and Sons, pp. 272-276. Extended version: *Technical Report IR-282*, Vrije Universiteit Amsterdam, Department of Mathematics and Computer Science, 1991.

LANGHOLM, T. (1988). Partiality, Truth and Persistence. *CSLI Lecture Notes, No. 15*. Stanford University, Stanford.

LEE, J. (1997). Design Rationale Systems: Understanding the Issues. In: *IEEE Expert*, **12**(3), pp. 78-85.

LEVY, A.Y., SAGIV, Y., and SRIVASTAVA, D. (1994). Towards efficient information gathering agents. In: *Software Agents*, proceedings of the AAAI 1994 spring symposium, AAAI Press, pp. 64-70.

LÖCKENHOFF, C., and MESSER, T. (1994). Configuration. In: BREUKER, J.A., and VELDE, W. VAN DE (Eds.), *The CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, chapter 9, pp. 197-212.

LOGAN, B., and SMITHERS, T. (1992). Creativity and Design as Exploration. *Technical Report*, DAI research paper no. 597. Revision of paper which appeared in: GERO, J.S., and MAHER, M.L. (Eds.), *modelling creativity and knowledge based creative design*., Lawrence Earlbaum, 1992.

LUCAS, P. (1996). *Structures in Diagnosis from theory to medical application*. PhD thesis, Free University, Amsterdam, The Netherlands.

MAHER, M.L., and DE SILVA GARZA, A.G. (1997). Case-Based Reasoning in Design. In: *IEEE Expert*, **12**(2), pp. 34-41.

MARCUS, S., and MCDERMOTT, J. (1989). SALT: A Knowledge-Acquisition Language for Propose and Revise Systems. In: *Journal of Artificial Intelligence*, **39**(1), pp. 1-37.

MARIR, F., and WATSON, I. (1994). Case-based reasoning: a categorized bibliography. In: *The Knowledge Engineering Review*, **9**(4), pp. 355-381.

MAZZA, C., FAIRCLOUGH, J., MELTON, B., DE PABLO, D., SCHEFFER, A., and STEVENS, R. (1994). *Software Engineering Standards*, Prentice Hall, Hertfordshire, UK.

MINTON, S. (1990). Quantitative Results Concerning the Utility of Explanation-Based Learning. In: *Artificial Intelligence*, **42**, pp. 363-392.

MORAN, T.P., and CARROLL, J.M. (1996). Overview of design rationale. In: MORAN, T.P., and CARROLL, J.M. (Eds.), *Design rationale: concepts, techniques, and use*. Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 1-19.

MOSTOW, J. (1989). Design by Derivational Analogy: Issues in the Automated Replay of Design Plans. In: *Artificial intelligence*, **40**, pp. 119-184.

MOTTA, E., and ZDRAHAL, Z. (1998). A principled approach to the construction of a task-specific library of problem solving components. In: (Gaines and Musen, 1998), pp. 20.

MOTTA, E., STUTT, A., ZDRAHAL, Z., O'HARA, K., and SHADBOLT, N. (1996). Solving VT in VITAL: a study in model construction and knowledge reuse. In: (Schreiber and Birmingham, 1996), pp. 333-371.

MULDER, M., TREUR, J., and FISHER, M. (1997). Agent Modelling in METATEM and DESIRE. In: SINGH, M.P., RAO, A.S., and WOOLDRIDGE, M.J. (Eds.), *Preproceedings of the 4th International Workshop on Agent Theories, Architectures, and Languages*. Providence, Rhode Island, USA, pp. 183-197.

MURRAY, K.J., and SHEPPARD, S.V. (1988). Knowledge-based simulation model specification. In: *Simulation*, **50**(3), pp. 112-119.

MUSEN, M.A. (1990). An editor for the conceptual models of interactive knowledge-acquisition tools. In: BOOSE, J.H., and GAINES, B.R. (Eds.), *The Foundations of Knowledge Acquisition*. Knowledge-Based Systems, Academic Press Limited, **4**, pp. 135-160.

NAYAK, P.P., and JOSKOWICZ, L. (1996). Efficient Compositional Modelling for Generating Causal Explanations. In: *Artificial Intelligence*, **83**(2), pp. 193-227.

NEWELL, A. (1982). The Knowledge Level. In: *Artificial Intelligence*, **18**, pp. 87-127.

NORMAN, D.A. (1994). How Might People Interact with Agents. In: *Communications of the ACM*, **37**(7), pp. 68-71.

NUMAOKA, C. (1996). Bacterial Evolution Algorithm for Rapid Adaptation. In: VAN DE VELDE, W., and PERRAM, J.W., (Eds.), *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, Lecture Notes in Artificial Intelligence, **1038**, Springer Verlag, pp. 139-148.

NWANA, H.S. (1996). Software agents: an overview. In: *The Knowledge Engineering Review*, Cambridge University Press, **11**(3), pp. 205-244.

OHSUGA, S. (1997). Strategic Knowledge Makes Knowledge Based Systems Truly Intelligent. In: CANDY, L., and HORI, K. (Eds.), *Proceedings of the First International Workshop on Strategic Knowledge and Concept Formation*. Lutchi Research Centre, pp. 1-24.

ORSVÄRN, K. (1996). *Knowledge Modelling with Libraries of Task Decopmosition Methods*. PhD Thesis, Royal Institute of Technology & Swedish Institute of Computer Science, SICS dissertation series 22, Kista, Sweden.

PAHL, G., and BEITZ, W. (1984). *Engineering Design*. Springer Verlag, New York. Originally: Konstruktionslehre, 1977, in German.

PEÑA-MORA, F., and VADHAVKAR, S. (1996). Design Rationale and Design Patterns in Reusable Software Design. In: GERO, J.S., and SUDWEEKS, F. (Eds.), *Artificial Intelligence in Design (AID'96)*, Kluwer Academic Publishers, pp. 251-268.

PIERRET-GOLBREICH, C. (1993). Task Model Perspective of Knowledge Engineering. In: *Proceedings of the 7th European Knowledge Acquisition Workshop (EKAW'93)*, Springer-Verlag.

PIERRET-GOLBREICH, C. (1994). Task Model, a framework for the design of Models of Expertise and their operationalization. In GAINES, B.R., and MUSEN, M.A. (Eds.), *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, KAW'94*, **2**, pp. 37/1-37/22. Calgary: SRDG Publications, Department of Computer Science, University of Calgary.

PIERRET-GOLBREICH, C., and TALON, X. (1997). Specification of Flexible Knowledge-Based Systems. In: PLAZA, E., and BENJAMINS, R. (Eds.), *Knowledge acquisition, modeling and management, Proceedings of the 10th European Workshop (EKAW'97)*, Springer Berlin, pp. 190-204.

PIRLEIN, T., and STUDER, R. (1995). An Environment for reusing ontologies within a knowledge engineering approach. In: *International Journal of Human-Computer Studies*, **43**, pp. 945-965.

POECK, K., FENSEL, D., LANDES, D., and ANGELE, J. (1996). Combining KARL and CRLM for designing vertical transportation systems. In: (Schreiber and Birmingham, 1996), pp. 435-467.

Pos, A., and AKKERMANS, H., (1996). 007: A system for automated model revision. In: JAVAR, A., LEHMANN, A., and MOLNAR, I. (Eds.), *Proceedings of the 10th European Simulation Multiconference (ESM'96)*, Budapest, Hungary, pp. 50-54.

POS, A., AKKERMANS, H., and STRAATMAN, R. (1997). Problem Solving for Re-design. In: PLAZA, E., and BENJAMINS, R. (Eds.), *Knowledge acquisition, modeling and management, Proceedings of the 10th European Workshop (EKAW'97)*, Springer Berlin, pp. 205-220.

PRESSMAN, R.S. (1997). *Software Engineering: A practitioner's approach.* Fourth Edition, McGraw-Hill Series in Computer Science, McGraw-Hill Companies Inc., New York.

PUERTA, A.R., EGAR, J.W., TU, S.W., and MUSEN, M.A. (1992). A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. In: *Knowledge Acquisition*, **4**, pp. 171-196.

RASMUSSEN, S., and BARRETT, C.L. (1995). Elements of a Theory of Simulation. In: *Proceedings of the European Conference on Artificial Life (ECAL'95)*, Lecture Notes in Computer Science, Springer Verlag.

REICH, Y. (1993). The development of Bridger: a methodological study of research on the use of machine learning in design. In: *Artificial Intelligence in Engineering*, **8**(3), pp. 217-231.

REICH, Y., and FENVES, S.J. (1995). A system that learns to design cable-stayed bridges. In: *Journal of structural Engineering*, ASCE, **21**(7), pp. 1090-1100.

REITER, R. (1987). A Theory of Diagnosis from First Principles. In: *Artificial Intelligence*, **32**, pp. 57-95.

REVISE (project) (1996). A Purpose Driven Method for Language Comparison. In: SHADBOLT, N., O'HARA, K., and SCHREIBER, A.TH. (Eds.), *Advances in Knowledge Acquisition. 9th European Knowledge Acquisition Workshop (EKAW'96)*. Lecture Notes in Artificial Intelligence, **1076**, pp. 66 - 81.

RICH, C.R., and SHROBE, H.E. (1986). Initial Report on a LISP Programmer's Apprentice. In: BARSTOW, D.R., SHROBE, H.E., and SANDEWALL, E. (Eds.), *Interactive Programming Environments*. B & Jo Enterprise Pte LTD, Singapore, pp. 443-463. Reprint from IEEE Transactions on Software Engineering, 1978, **4**(6), pp. 456-467.

RIEL, A.J. (1996). *Object-Oriented Design Heuristics*. Addison Wesley Publishing Company, Reading Massechusetts.

RIST, R.S. (1995). Program structure and design. In: *Cognitive Science*, **19**, pp, 507-562.

SAGE, A.P., and PALMER, J.D. (1990). *Software Systems Engineering*. John Wiley and Sons, New York.

SCHÖN, D.A. (1983). *The reflective practitioner: how professionals think in action*. Temple Smith, London.

SCHREIBER, A.TH., and BIRMINGHAM, W.P. (1996) (Eds.). *Special Issue on Sisyphus-VT*. In: *International Journal of Human-Computer Studies (IJHCS)*, **44**.

SCHREIBER, A.TH., and TERPSTRA, P. (1996). Sisyphus-VT: A CommonKADS solution. In: (Schreiber and Birmingham, 1996), pp. 303-332.

SCHREIBER, A.TH., WIELINGA, B.J., AKKERMANS, J.M., VELDE, W. VAN DE, and HOOG, R. de (1994). CommonKADS: A comprehensive methodology for KBS development. In: *IEEE Expert*, **9**(6).

SCHUBERT, F. (1997). A reflective architecture for an adaptable object-oriented operating system based on C++. In: *Proceedings of the ECOOP'97 workshop on object-orientation and operating systems*, Springer Verlag, Lecture notes in Computer Science, **1357**.

SHADBOLT, N., MOTTA, E., and ROUGE, A. (1993). Constructing Knowledge-Based Systems. In: *IEEE Software*, November, pp. 34-38.

SHAW, M.L.G., and GAINES, B.R. (1995). Knowledge and Requirements Engineering. In: (Gaines and Musen, 1995), pp. 44/1-44/22.

SHOHAM, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, **60**, pp. 51-92.

SLOOF, M. (1998). Automated Modelling of Physiological Processes during Postharvest Distribution of Agricultural Products. In: *Artificial Intelligence Review Journal*, special volume on Artificial Intelligence for Biology and Agriculture, **12**, pp. 39-70.

SMITH, D.R. (1990). KIDS: A Semi-Automatic Program Development System. In: *IEEE Transactions on Software Engineering*, Special Issue on Formal Methods, **16**, pp. 1024-1043.

SMITH, D.R. (1991). KIDS - A Knowledge-Based Software Development System. In: Lowry, M., and McCartney, R. (Eds.), *Automating Software Design*, AAAI/MIT Press, pp. 483-514.

SMITHERS, T. (1992). Design as Exploration: Puzzle-Making and Puzzle-Solving. In: *Proceedings of the workshop on exploration-based models of design and search-based models of design at second international conference on AI in Design*.

SMITHERS, T. (1996). On Knowledge Level Theories of Design Process. In: GERO, J.S., and SUDWEEKS, F. (Eds.), *Artificial Intelligence in Design (AID'96)*, Kluwer Academic Publishers, pp. 561-579.

SMITHERS, T., and TROXELL, W. (1990). Design is intelligent behaviour, but what's the formalism? In: *AIEDAM*, **4**(2), pp. 89-98.

SMITHERS, T., CORNE, D., and ROSS, P. (1994). On Computing Exploration and Solving Design Problems. In: *Proceedings of the IFIP WG5.2 international workshop on formal methods for CAD 1993*, published by North Holland.

SOMMERVILLE, I., and RODDEN, T. (1994). Requirements Engineering for Cooperative Systems. *Technical Report CSCW/1/1994*, University of Lancaster.

SOMMERVILLE, I., and SAWYER, P. (1997). *Requirements Engineering: a good practice guide*. John Wiley & Sons, Chicester, England.

STEIER, D. (1991). Automating Algorithm Design within a General Architecture for Intelligence. In: LOWRY, M.R., and MCCARTNEY, R.D. (Eds.), *Automating Software Design*, AAAI Press, pp. 577-602.

STRAATMAN, R. (1997). Kids for Kads. In. PLAZA, E., and BENJAMINS, R. (Eds). *Proceedings of the 10th European Workshop on Knowledge Acquisition, Modelling, and Management (EKAW'97)*. Sant Feliu de Guixols, Catalania, Lecture Notes in Artificial Intelligence, **1319**, Springer Verlag, pp. 371-376.

STRELNIKOV, Y.N., and DMITREVICH, G.D. (1991). Formal description and comparison of interactive design strategies. In: *Artificial Intelligence in Engineering*, **6**(4), pp. 186-195.

STROULIA, E., and GOEL, A.K. (1994a). Reflective, Self-Adaptive Problem Solvers. In: STEELS, L., SCHREIBER, G., and VAN DE VELDE, W. (Eds.), *A future for knowledge acquisition, proceedings of the 1994 European Conference on Knowledge Acquisition (EKAW'94)*, Lecture Notes in Artificial Intelligence, **867**, Springer-Verlag, pp. 394-413.

STROULIA, E., and GOEL, A.K. (1994b). Learning Problem-Solving Concepts by Reflecting on Problem Solving. In: BERGADANO, and DE RAEDT, L. (Eds.), *Proceedings of ECML-94*, Springer-Verlag, pp. 287-306.

STUTT, A., and MOTTA, E. (1995). Recording the design decisions of knowledge engineers to facilitate re-use of design models. In: (Gaines and Musen, 1995), pp. 33/1-33/19.

SUMI, Y. (1997). Supporting the Acquisition and Modelling of Requirements in Software Design. In: CANDY, L., and HORI, K. (Eds.), *Proceedings of the First International Workshop on Strategic Knowledge and Concept Formation*, Lutchi Research Centre, pp. 205-216.

SYCARA, K., and ZENG, D. (1996). Multi-agent integration of information gathering and decision support. In: WAHLSTER, W. (Ed), *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96)*, Wiley and Sons, pp. 549-553.

TAKAHASHI, M., OONO, J., SAITOH, K., and MATSUMOTO, S. (1995). Reusing makes it easier: Manufacturing Process Design by CBR with KnowledgeWare. In: *IEEE Expert*, pp. 74-80.

TAKEDA, H., VEERKAMP, P.J., TOMIYAMA, T., and YOSHIKAWA, H. (1990), Modelling Design Processes. In: *AI Magazine*, **11**(4), 37-48.

TALON, X., and PIERRET-GOLBREICH, C. (1996*a*). TASK, a framework for the different steps of a KBS construction. In: Proceedings of the *6th European Workshop on Knowledge Engineering Methods and Languages* (KEML'96), Gif sur Yvette.

TALON, X., and PIERRET-GOLBREICH, C. (1996*b*). TASK: from the specification to the implementation. In: *8th IEEE International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 80-88.

TEIJE, A. TEN, and HARMELEN, F. VAN (1994). An extended spectrum of logical definitions for diagnostic systems. In: *Proceedings of DX-94 Fifth International Workshop on Principles of Diagnosis*, New Paltz, New York, pp. 334-342.

TEIJE, A. TEN, and Harmelen, F. VAN (1996). Using reflection techniques for flexible problem solving. In: *Future Generation Computer Systems*, **12**, special issue Reflection and Meta-level AI Architectures, pp. 217-234.

TEIJE, A. TEN, HARMELEN, F. VAN, SCHREIBER, A.Th., WIELINGA, B.J. (1998). Construction of problem-solving methods as parametric design. *International Journal of Human-Computer Studies*, Special issue on problem-solving methods, **49**(4).

TEITELMAN, W. (1986a). A Display-Oriented Progammer's Assistant. In: BARSTOW, D.R., SHROBE, H.E., and SANDEWALL, E. (Eds.), *Interactive Programming Environments*. B & Jo Enterprise Pte LTD, Singapore, pp. 240-287 (reprint).

TEITELMAN, W. (1986b). Automated Programmering: The Programmer's Assistant. In: BARSTOW, D.R., SHROBE, H.E., and SANDEWALL, E. (Eds.), *Interactive Programming Environments*. B & Jo Enterprise Pte LTD, Singapore, pp. 232-239. Reprint from AFIPS: Fall Joint Computer Conference Proceedings, 1972, AFIPS, **41**, pp. 917-921.

TERVEEN, L.G., and MURRAY, L.T. (1996). Helping Users Program Their Personal Agents. In: TAUBER, M.J. (Ed.), *Proceedings of the conference of Human Factors in Computing Systems (CHI'96)*, pp. 355-361.

THAM, K W., and GERO, J.S. (1992). PROBER – A design system based on design prototypes. In: GERO, J.S. (Ed.), *Artificial Intelligence in Design (AID'92)*, Kluwer, Dordrecht, pp. 657-675.

TOMIYAMA, T., and YOSHIKAWA, H. (1987). Extended General Design Theory. In: H. YOSHIKAWA and E.A. WARMAN (Eds.), *Proceedings of the IFIP WG 5.2 Working Conference on Design Theory for CAD*, North-Holland, pp. 95-125.

TREUR, J. (1989). A logical analysis of design tasks for expert systems. In: *International Journal of Expert Systems*, **2**, 233-253.

TREUR, J. (1993). Heuristic reasoning and relative incompleteness. In: *Journal of Approximate Reasoning*, **8**, pp. 51-87.

TREUR, J. (1994). Temporal Semantics of Meta-Level Architectures for Dynamic Control of Reasoning. In: FRIBOURG, L., and TURINI, F. (Eds.), *Logic Program Synthesis and Transformation-Meta-Programming in Logic, Proceedings of the Fourth International Workshop on Meta-Programming in Logic (META'94)*. Springer Verlag, Lecture Notes in Computer Science, **883**, pp. 353-376.

TREUR, J., AKKERMANS, J.M., MARS, N.J.I., and WIELINGA, B.J. (1992). Research proposal: Evolutionary Design in Knowledge-Based Systems. NWO/SION Project Number 612-322-316, The Netherlands.

TREUR, J., and WETTER Th. (1993) (Eds.). *Formal Specification of Complex Reasoning Systems*, Ellis Horwood, 282 pp.

UMEDA, Y., and TOMIYAMA, T. (1997). Functional Reasoning in Design. In: *IEEE Expert*, **12**(2), pp. 42-48.

VANWELKENHUYSEN, J. (1995). Embedding non-functional requirements analyses in conceptual knowledge system designs. In: (Gaines and Musen, 1995), pp. 45/1-45/15.

VANWELKENHUYSEN, J., and MIZOGUCHI, R. (1995). Workplace-Adapted Behaviors: Lessons Learned for Knowledge Reuse. In: *Proceedings of Second International Conference on Building and Sharing Very Large-Scale Knowledge Bases*, Enschede, Netherlands.

VESCOVI, M., and IWASAKI, Y. (1993), Device design as functional and structural refinement. In: FALTINGS, B. (Ed.), *Working Notes of the IJCAI'93 Workshop on AI in Design* , Chambéry, pp. 55-60.

VLIET, J.C. VAN (1993). *Software Engineering: Principles and Practice*. John Wiley & Sons, Chicester.

Voß, A. (1997). Case reusing systems - survey, framework and guidelines. In: *The Knowledge Engineering Review*, **12**(1), pp. 59-89.

Voß, A., and Oxman, R. (1996). A stury od case adaptation systems. In: GERO, J.S., and SUDWEEKS, F. (Eds.), *Artificial Intelligence in Design (AID'96)*, Kluwer Academic Publishers, pp. 173-189.

WANG, Q., RAO, M., and ZHOE, J. (1995). Intelligent systems approach to conceptual design. In: *International journal of inteillgent systems*, **10**, pp. 259-293.

WATERS, R.C. (1986). The Programmer's Apprentice: Knowledge-Based Progam Editing. In: BARSTOW, D.R., SHROBE, H.E., and SANDEWALL, E. (Eds.), *Interactive Programming Environments*. B & Jo Enterprise Pte LTD, Singapore, pp. 464-486. Reprint from IEEE Transactions on Software Engineering, 1982, **8**(1), pp. 1-12.

WATSON, I., and MARIR, F. (1994). Case-based reasoning: A review. In: *The Knowledge Engineering Review*, **9**(4), pp. 327-354.

WIELINGA, B.J., and SCHREIBER, A.Th. (1997). Configuration design problem solving. In: *IEEE Expert*, **12**(2), Special issue on AI and design, pp. 49-56.

WIELINGA, B.J., SCHREIBER, A.TH., and BREUKER, J.A. (1992). KADS: a modelling approach to knowledge engineering. In: *Knowledge Acquisition*, **4**, pp. 5-53.

WIERINGA, R.J. (1996). *Requirements Engineering: Frameworks for Understanding*. Wiley and Sons.

WILLIAMS, B.C., and NAYAK, P.P. (1996). A Model-Based Approach to Reactive Self-Configuring Systems. In: *Proceedings of the AAAI-96*, **2**, pp. 971-978.

WINSOR, J., and MACCALLUM, K. (1994). A review of functionality modelling in design. In: *The Knowledge Engineering Review*, **9**(2), pp. 163-199.

WOOLDRIDGE, M.J., and JENNINGS, N.R. (1995). Intelligent Agents: Theory and Practice. In: *Knowledge Engineering Review*, **10**(2), pp. 115-152.

ZAFF, B.S., MCNEESE, M.D., and SNYDER, D.E. (1993). Capturing multiple perspectives: a user centered approach to knowledge and design acquisition. In: *Knowledge Acquisition*, **5**, pp. 79-116.

ZDRAHAL, Z., and MOTTA, E. (1995). An In-Depth Analysis of Propose & Revise Problem Solving Methods. In: (Gaines and Musen, 1995), pp. 38/1-38/20.

Part **VI**

# Appendices

In this part additional details of the generic design model and the re-design model for compositional systems are described. The generic design model is described in Chapter 6 at an abstract level; in Appendix A a number of elements are described in more detail. The re-design model for compositional systems is described in Chapters 7, 8 and 9, in which a number of sub-processes are not described in further detail. In Appendix B, these sub-processes are described in more detail.

# A Additional Descriptions of a Generic Model for Design

The generic model of design is described in Chapter 6. A number of aspects of the process composition and knowledge composition related to the generic design model are not discussed in Chapter 6; these aspects are discussed in this appendix.

In Section A.1 additional information types related to the process design are described. In Section A.2 additional aspects of the process DOD manipulation are described, and in Section A.3 additional aspects of the process RQS manipulation are described.

## A.1 Information types related to the process Design

In this section information types are described which are related to the information types design process objectives, design process evaluation, overall design strategy , and manipulation process evaluation.

### A.1.1 Design process objectives

The information type design process objectives consists of two information types: process objectives and qualified process objectives, as shown in Figure A.1. This distinction is similar to the distinction within design requirements between requirements, and qualified requirements. The information types process objectives and qualified process objectives can be extended.

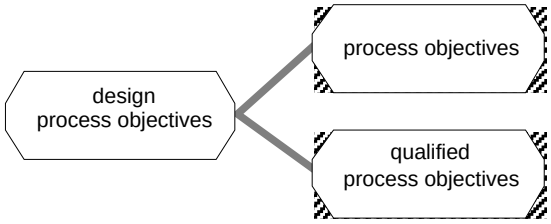


**Figure A.1** Partial view on information type design process objectives.

The generic relation defined in the information type process objectives is:

**relations**

| | |
|---|---|
| is_process_objective: | process_objective_name * process_info_element_expression; |

The sort process info element expression contains the meta-descriptions of design process statements (e.g., which manipulation process is supposed to do what), on which logical operators are defined to be able to formulate expressions. Generic design process statements are given below:

**functions**

| | |
|---|---|
| is_RQS_to_be_used,<br>is_RQS_to_be_refined: | RQS_name → process_info_element_expression; |
| is_DOD_to_be_used,<br>is_DOD_to_be_refined: | DOD_name → process_info_element_expression; |

The terms is RQS to be used and is DOD to be used express that a specific RQS and DOD have to be used during the design process. The terms is RQS to be refined and is DOD to be refined express that a specific RQS and DOD have to be refined during the design process (and also used).

The generic relation defined in the information type qualified process objectives is:

**relations**

| | |
|---|---|
| is_qualified_process_objective: | qualified_process_objective_name * |
| | process_objective_qualification * |
| | process_objective_name_tuple; |

The relation is qualified process objective denotes which objectives with specific qualifications are to be realised by the design process.

### A.1.2 Design process evaluation

The information type design process evaluation consists of two information types: process evaluation and qualified process evaluation, as shown in Figure A.2. The information type process evaluation contains an evaluation of process objectives, and the information type qualified process evaluation contains an evaluation of qualified process objectives. These two information types can be extended.

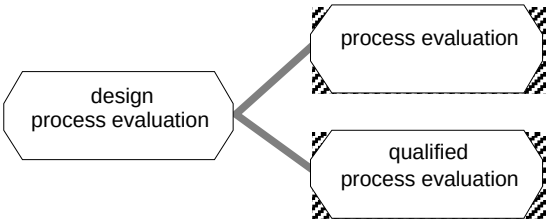


**Figure A.2** Partial view on information type design process evaluation.

Generic relations defined in the information type design process evaluation results are:

**relations**

| | |
|---|---|
| is_satisfied_process_objective, | |
| is_violated_process_objective, | |
| is_evaluated_process_objective: | process_objective_name; |
| is_reached_qualified_process_objective, | |
| is_unreached_qualified_process_objective, | |
| is_evaluated_qualified_process_objective: | qualified_process_objective_name; |

These relations express whether a process objective has been satisfied or violated (but not both), whether a process objective has been evaluated, whether a qualified process objective has been reached or not (but not both), and whether a qualified process objective has been evaluated.

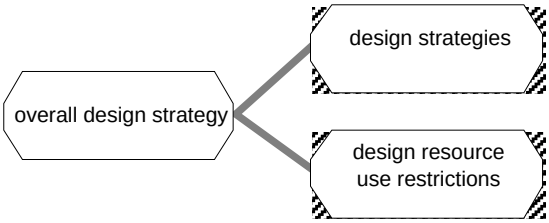


**Figure A.3** Partial view on information type overall design strategy.

### A.1.3 Overall design strategy

The information type overall design strategy consists of two information types: design strategies and design resource use restrictions, as shown in Figure A.3. The information type design strategies contains information on strategies for manipulation processes, the information type

design resource use restrictions contains information on restrictions on design resources such as time and funding. Both information types can be extended.

The generic relation defined in the information type design strategies is:

**relations**

| | |
|---|---|
| is_design_strategy: | design_strategy_name * |
| | manipulation_process_property_expression; |

The sort manipulation process property expression denotes logical expressions on properties of manipulation processes. Generic manipulation process properties are:

**functions**

| | |
|---|---|
| is_DOD_to_be_revised_to_satisfy, | |
| is_DOD_to_be_refined_to_satisfy: | DOD_name * |
| | RQS_name |
| | → manipulation_process_property; |
| is_RQS_to_be_revised, | |
| is_RQS_to_be_refined: | RQS_name |
| | → manipulation_process_property; |

The terms is DOD to be revised to satisfy and is DOD to be refined to satisfy express that a specific DOD is to be manipulated so that a DOD will result which satisfies a specific RQS; these terms are used by the process DOD Manipulation. The terms is RQS to be revised and is RQS to be refined express that a specific RQS is to be manipulated; these terms are used by the process RQS Manipulation.

The generic relation defined in the information type design resource use restrictions is:

**relations**

| | |
|---|---|
| imposes_design_resource_use_restriction: | design_strategy_name * |
| | restriction * |
| | design_resource; |

The sort restriction denotes restrictive expressions, the generic expressions at most, exactly, and at least are available from the generic design model. The sort design resource denotes a design resource, for example funding and time.

### A.1.4 Manipulation process evaluation

The information type manipulation process evaluation consists of two information types: RQSM process evaluation and DODM process evaluation as shown in Figure A.4. Both information types contain information on the relation between a manipulation process and the given overall design strategy and both can be extended.

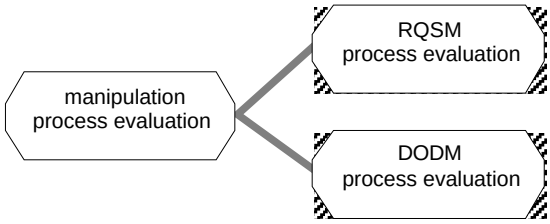


**Figure A.4** Partial view on information type manipulation process evaluation.

The generic relations defined in the information types RQSM process evaluation and DODM process evaluation can be explained together:

**relations**

| | |
|---|---|
| has_strategic_DODM_result, | |
| has_strategic_RQSM_result: | design_strategy_name * |
| | manipulation_process_property_expression * |
| | strategic_manipulation_result; |

*(continued)*

| | |
|---|---|
| has_DODM_resource_use_result,<br>has_RQSM_resource_use_result: | design_strategy_name *<br>restriction *<br>design resource *<br>strategic_manipulation_result; |

The relations has strategic DODM result and has strategic RQSM result express which results have been achieved for parts of a specific overall design strategy. The sort strategic manipulation result contains: partial success, complete success, partial failure, complete failure, and no action required as generic objects.

## A.2 Additional description of DOD manipulation

Additional aspects of the process composition and knowledge composition of DOD manipulation are described in this section. In Section A.2.1. the interfaces of the sub-processes of DOD manipulation are described. Section A.2.2 describes the information links within DOD manipulation in some detail. Section A.2.3 describes additional information types related to DOD manipulation.

### A.2.1 Description of the interfaces of sub-processes of DODM

The input and output information types in the interfaces of the processes described in Table 6.3 are elaborated below:

- The process DOD Modification has, as input, results of searching the DOD modification state history ( DOD modification state history search results), overall design strategy (overall design strategy), results of searching the DOD assessment history (DOD assessment history search results), results of searching the DOD history (DOD history search results), results of searching the RQS history (RQS history search results), results on the success of replacing the current DOD (current DOD replacement results), and the contents of the current DOD (current DOD contents). DOD modification generates information on the progress of the modification process (DOD modification progress), actions for the manipulation process (current manipulation action), queries on DOD modification state history ( DOD modification state history queries), assessments of design object descriptions (DOD assessment), queries on DOD assessment history (DOD assessment history queries), queries on requirement qualification sets (RQS history queries), foci for deductive refinement of design object descriptions (DOD refinement focus), a focus within the current DOD (current DOD focus), modifications for the current DOD (current DOD modification), queries on DOD history information (DOD history query specification), and request for replacement of the current DOD (current DOD replacement request).
- The process DODM History Maintenance has, as input, information on the progress of the modification process (DOD modification progress), overall design strategy (overall design strategy) assessments of design object descriptions (DOD assessment), queries on DOD assessment history (DOD assessment history queries), design object descriptions (DOD), requirement qualification sets (RQS), queries on RQS history (RQS history queries), queries on the DOD history (DOD history queries), and queries on DOD modification state history (DOD modification state history queries), requests for the replacement of the DOD currently in focus (current DOD replacement request), and the contents of a DOD which needs to be stored (current DOD contents). DODM History Maintenance produces results of searching the DOD modification state history ( DOD modification state history search results), results of searching the DOD assessment history (DOD assessment history search results), results of searching the RQS history (RQS history search results), results of searching the DOD history (DOD history search results), results on the success of replacing the current DOD (current

DOD replacement results), and the contents of the new, current, DOD (new current DOD contents).
- The process deductive DOD refinement has, as input, information on basic design object (basic design object information) and, as output, information on design objects: both basic and derived design objects (design object information).
- The process current DOD maintenance has, as input and as output, information on design objects (design object information).

### A.2.2 Information exchange within DODM

Within the component DOD Manipulation seven mediating links and twenty private links are defined, as shown in Figure 6.9:

- The mediating link overall design strategy to history transfers overall design strategy from the input interface of DOD Manipulation to the input interface of DODM History Maintenance.
- The mediating link overall design strategy transfers overall design strategy from the input interface of DOD Manipulation to the input interface of DOD Modification.
- The mediating link initial DOD transfers DOD from the input interface of DOD Manipulation to the input interface of DODM History Maintenance.
- The mediating link RQS transfers RQS from the input interface of DOD Manipulation to the input interface of DODM History Maintenance.
- The private link DOD modification progress transfers DOD modification progress from the output interface of DOD Modification to the input interface of DODM History Maintenance.
- The private link DOD modification state history queries transfers DOD modification state history queries from the output interface of DOD Modification to the input interface of DODM History Maintenance.
- The private link current DOD replacement request transfers current DOD replacement request from the output interface of DOD Modification to the input interface of DODM History Maintenance.
- The private link DOD assessment to history transfers DOD assessment from the output interface of DOD Modification to the input interface of DODM History Maintenance.
- The private link DOD assessment history queries transfers DOD assessment history queries from the output interface of DOD Modification to the input interface of DODM History Maintenance.
- The private link RQS history queries transfers RQS history queries from the output interface of DOD Modification to the input interface of DODM History Maintenance.
- The private link DOD history queries transfers DOD history queries from the output interface of DOD Modification to the input interface of DODM History Maintenance.
- The private link DOD refinement focus transfers DOD refinement focus from the output interface of DOD Modification to targets on design requirement information in the input interface of deductive DOD refinement on the basis of an explicit mapping between these information types.
- The private link current DOD focus transfers current DOD focus from the output interface of DOD Modification to assumptions on design object information in the input interface of current DOD maintenance on the basis of an explicit mapping between these information types.
- The private link DOD modifications transfers current DOD modification from the output interface of DOD Modification to assumptions on design object information in the input interface of current DOD maintenance on the basis of an explicit mapping between these information types.
- The private link DOD modification state history search results transfers DOD modification state history search results from the output interface of DODM History Maintenance to the input interface of DOD Modification.

- The private link DOD assessment history search results transfers DOD assessment history search results from the output interface of DODM History Maintenance to the input interface of DOD Modification.
- The private link RQS history search results transfers RQS history search results from the output interface of DODM History Maintenance to the input interface of DOD Modification.
- The private link DOD history search results transfers DOD history search results from the output interface of DODM History Maintenance to the input interface of DOD Modification.
- The private link current DOD replacement results transfers current DOD replacement results from the output interface of DODM History Maintenance to the input interface of DOD Modification.
- The private link new current DOD transfers new current DOD contents from the output interface of DODM History Maintenance to assumptions on design object information in the input interface of current DOD maintenance on the basis of an explicit mapping between these information types.
- The private link current DOD to be stored transfers epistemic information on design object information from the output interface of current DOD maintenance to current DOD contents in the input interface of DODM History Maintenance on the basis of an explicit mapping between these information types.
- The private link current DOD to be analysed transfers epistemic information on design object information from the output interface of current DOD maintenance to current DOD contents in the input interface of DOD Modification.
- The private link basic design object information transfers basic design object information from the output interface of current DOD maintenance to the input interface of deductive DOD refinement.
- The private link design object information transfers design object information from the output interface of deductive DOD refinement to the input interface of current DOD maintenance.
- The mediating link DOD transfers DOD from the output interface of DODM History Maintenance to the output interface of DOD Manipulation.
- The mediating link DOD assessment transfers DOD assessment from the output interface of DOD Modification to the output interface of DOD Manipulation.
- The mediating link DODM process evaluation transfers DODM process evaluation from the output interface of DOD Modification to the output interface of DOD Manipulation.

### A.2.3 Knowledge composition related to DOD Manipulation

Additional information types are described, which are related to the information types DOD modification progress, DOD modification state history, DOD assessment history, and DOD history. The information types related to RQS history, although used in interfaces within DOD Manipulation, are described in Section A.3.3.

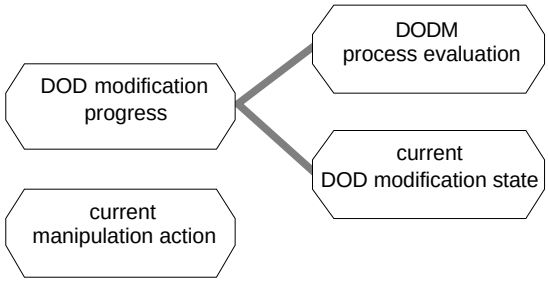


**Figure A.5** Information types related to DOD modification progress.

**DOD modification progress.** The information type DOD modification progress, see Figure A.5, describes the current state of the modification process, and the information available on

success and/or failure of the modification process. This information type consists of current DOD modification state and DODM process evaluation. The information type DODM process evaluation is described in Section A.1. The information type current manipulation action is related to the information type DOD modification progress: manipulation actions influence the sub-processes within the process DOD manipulation.

The generic relation defined in the information type current DOD modification state is:

**relations**
is_current_DOD_modification_state_value:
DOD_modification_state_attribute *
DOD_modification_state_attribute_value;

This relation describes the contents of the current modification state, without attaching an identifier (i.e., the sort DOD modification state). Within the DODM History Maintenance an identifier is attached, as well as additional temporal and persistent information (e.g., the final DOD).

The generic relation defined in the information type current manipulation action is:

**relations**
is_current_manipulation_action: manipulation_action;

This relation describes which manipulation action (or actions) is (or are) to be performed next. This influences activation of information links and sub-processes within the process DOD Manipulation.

**DOD modification state history.** The information type DOD modification state history is employed to describe information related to modification 'steps'. It is related to a number of information types, as shown in Figure A.6. The information type DOD modification state history consists of two information types: temporal DOD modification state information, and persistent DOD modification state information. The latter information type consists of two information types: DOD modification state information, and DOD modification state sequence information. Three other information types are conceptually related to DOD modification state history: DOD modification state history queries, DOD modification state history search results, and DOD modification state history query results. The information type DOD modification state history search results refers to DOD modification state history and DOD modification state history query results.

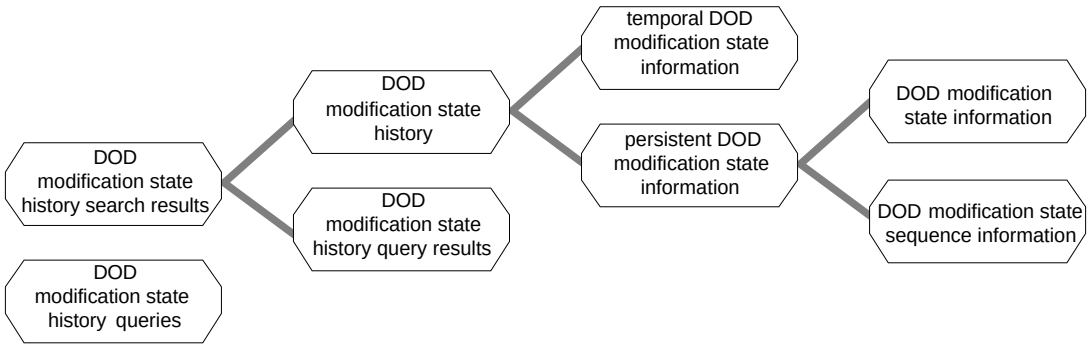


**Figure A.6** Information types related to DOD modification state history.

Generic relations defined in the information type temporal DOD modification state information are:

**relations**
is_newest_DOD_modification_state,
is_initial_DOD_modification_state: DOD_modification_state;

These relations denote which DOD modification state is the latest, newest (which is relative to modification state steps taken by DOD Modification), and which modification state is the initial modification state (which is relative to each 'design problem setting').

The generic relation defined in the information type DOD modification state information is:

```
relations
    has_DOD_modification_state_value:    DOD_modification_state *
                                         DOD_modification_state_attribute *
                                         DOD_modification_state_attribute_value;
```

This relation can be used to describe which overall design strategy is related to a modification state, which modifications are related to a modification state, on which DOD the modifications are based, which DOD is the result of this modification step, etc.

Generic relations defined in the information type DOD modification state sequence information are:

```
relations
    has_preceding_DOD_modification_state,
    has_succeeding_DOD_modification_state:    DOD_modification_state *
                                              DOD_modification_state;
    is_first_DOD_modification_state:          DOD_modification_state;
```

These relations chain together modification states, and define which modification state was the very first modification state in the DODM history.

The information types DOD modification state history queries and DOD modification state history query results contain relations with which queries can be formulated on the modification state history, and relations with which the results of the queries can be expressed.

**DOD assessment history.** The information type DOD assessment history is employed to retain information on assessments of design object descriptions (comparisons among design object descriptions, and evaluations of design object descriptions on the basis of sets of qualified requirements). It is related to a number of information types, as shown in Figure A.7. The information type DOD assessment history consists only of the information type DOD assessment, as it is not necessary to retain temporal information. Three other information types are conceptually related to DOD assessment history: DOD assessment history queries, DOD assessment history search results, and DOD assessment history query results. The information type DOD assessment history search results refers to DOD assessment history and DOD assessment history query results.

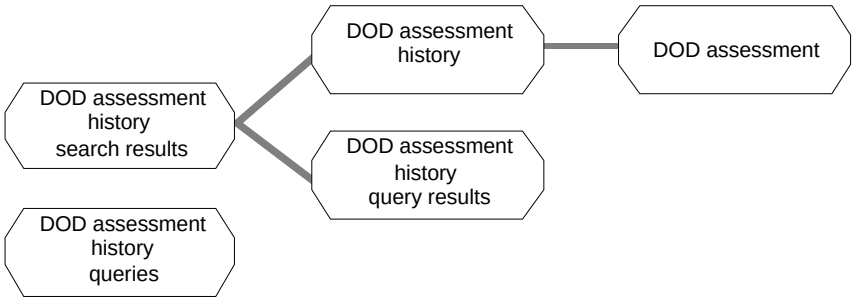


**Figure A.7** Information types related to DOD assessment history.

The information type DOD assessment is described in Section 6.1.2. The information types DOD assessment history queries and DOD assessment history query results contain relations with which queries can be formulated on the DOD assessment history, and relations with which the results of the queries can be expressed.

**DOD history.** The information type DOD history is employed to retain information on design object descriptions. It is related to a number of information types, as shown in Figure A.8. The information type DOD history, consists of two information types: temporal DOD information and persistent DOD information. The latter information type consists of two information types: DOD and DOD sequence information. Three other information types are conceptually related to DOD

history: DOD history queries, DOD history search results, and DOD history query results. The information type DOD history search results refers to DOD history and DOD history query results.

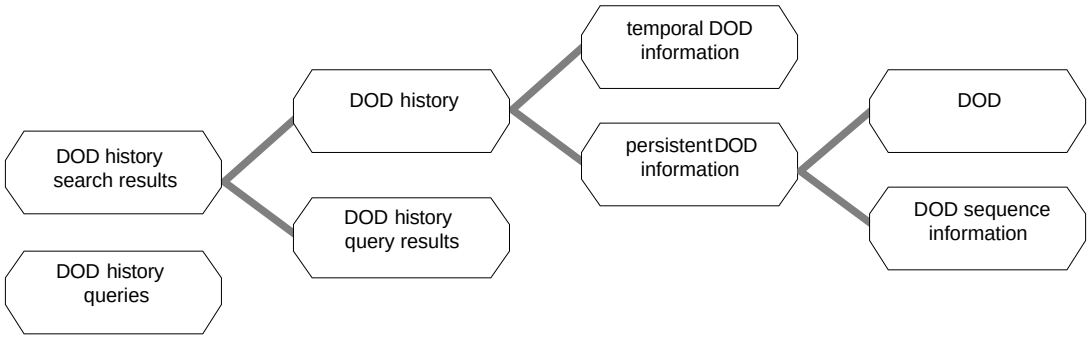


**Figure A.8** Information types related to DOD history information.

The information type DOD is described in Section 6.1. The generic relations defined in the information type temporal DOD information are:

**relations**
is_newest_DOD,
is_initial_DOD: DOD_name;

These relations denote which DOD is the latest, newest (i.e., last stored via a current DOD contents, or as an initial DOD), and which DOD is the initial DOD (which is relative to each 'design problem setting').

The generic relations define in the information type DOD sequence information are:

**relations**
has_previous_DOD,
has_next_DOD: DOD_name * DOD_name;
is_first_DOD: DOD_name;

These relations chain together design object descriptions, and define which design object description was the very first design object description in the DODM history.

The information types DOD history queries, and DOD history query results contain relations with which queries can be formulated on the DOD history, and relations with which the results of the queries can be expressed.

## A.3 Additional description of RQS manipulation

Additional aspects of the process composition and knowledge composition of RQS manipulation are described in this section. In Section A.3.1. the interfaces of the sub-processes of RQS manipulation are described. Section A.3.2 describes the information links within RQS manipulation in some detail. Section A.3.3 describes additional information types related to RQS manipulation.

### A.3.1 Description of the interfaces of sub-processes of RQSM

The input and output information types in the interfaces of the processes described in Table 6.4 are elaborated below:

- The process RQS Modification has, as input, results of searching the RQS modification state history ( RQS modification state history search results), overall design strategy (overall design strategy), results of searching the DOD assessment history (DOD assessment history search results), results of searching the RQS assessment history (RQS assessment history search results), results of searching the RQS history (RQS history search results), results on the success of replacing the current RQS (current RQS replacement results), and the contents

of the current RQS (current RQS contents). RQS modification generates information on the progress of the modification process (RQS modification progress), actions for the manipulation process (current manipulation action), queries on RQS modification state history ( RQS modification state history queries), assessments of sets of qualified requirements (RQS assessment), queries on RQS assessment history (RQS assessment history queries), queries on DOD assessment history (DOD assessment history queries), foci for deductive refinement of requirement qualification sets (RQS refinement focus), a focus within the current RQS (current RQS focus), modifications for the current RQS (current RQS modification), queries on RQS history information (RQS history query specification), and request for replacement of the current RQS (current RQS replacement request).

- The process RQSM History Maintenance has, as input, information on the progress of the modification process (RQS modification progress), assessments of requirement qualification sets (RQS assessment), given requirement qualification sets (RQS), queries on the RQS history, RQS assessment history, DOD assessment history, and historical RQS modification states (RQS history queries, RQS assessment history queries, DOD assessment history queries, and RQS modification state history queries), requests for the replacement of the RQS currently in focus (current RQS replacement request), and the contents of a RQS which needs to be stored (current RQS contents). RQSM History Maintenance produces results of searching the RQS modification state history ( RQS modification state history search results), results of searching the RQS assessment history (RQS assessment history search results), results of searching the DOD assessment history (DOD assessment history search results), results of searching the RQS history (RQS history search results), results on the success of replacing the current RQS (current RQS replacement results), and the contents of the new, current, RQS (new current RQS contents).
- The process deductive RQS refinement has, as input, information on basic design requirements (basic design requirement information) and, as output, information on design requirements: both basic and derived design requirements (design requirement information).
- The process current RQS maintenance has, as input and as output, information on design requirements (design requirement information).

### A.3.2 Information exchange within RQSM

Within the component RQS Manipulation eight mediating links and twenty private links are defined, as shown in Figure 6.13:

- The mediating link overall design strategy to history transfers overall design strategy from the input interface of RQS Manipulation to the input interface of RQSM History Maintenance.
- The mediating link overall design strategy transfers overall design strategy from the input interface of RQS Manipulation to the input interface of RQS Modification.
- The mediating link initial RQS transfers RQS from the input interface of RQS Manipulation to the input interface of RQSM History Maintenance.
- The mediating link DOD assessment transfers DOD assessment from the input interface of RQS Manipulation to the input interface of RQSM History Maintenance.
- The private link RQS modification progress transfers RQS modification progress from the output interface of RQS Modification to the input interface of RQSM History Maintenance.
- The private link RQS modification state history queries transfers RQS modification state history queries from the output interface of RQS Modification to the input interface of RQSM History Maintenance.
- The private link current RQS replacement request transfers current RQS replacement request from the output interface of RQS Modification to the input interface of RQSM History Maintenance.
- The private link RQS history queries transfers RQS history queries from the output interface of RQS Modification to the input interface of RQSM History Maintenance.

- The private link RQS assessment to history transfers RQS assessment from the output interface of RQS Modification to the input interface of RQSM History Maintenance.
- The private link RQS assessment history queries transfers RQS assessment history queries from the output interface of RQS Modification to the input interface of RQSM History Maintenance.
- The private link DOD assessment history queries transfers DOD assessment history queries from the output interface of RQS Modification to the input interface of RQSM History Maintenance.
- The private link RQS refinement focus transfers RQS refinement focus from the output interface of RQS Modification to targets on design requirement information in the input interface of deductive RQS refinement on the basis of an explicit mapping between these information types.
- The private link current RQS focus transfers current RQS focus from the output interface of RQS Modification to assumptions on design requirement information in the input interface of current RQS maintenance on the basis of an explicit mapping between these information types.
- The private link RQS modifications transfers current RQS modification from the output interface of RQS Modification to assumptions on design requirement information in the input interface of current RQS maintenance on the basis of an explicit mapping between these information types.
- The private link RQS modification state history search results transfers RQS modification state history search results from the output interface of RQSM History Maintenance to the input interface of RQS Modification.
- The private link RQS history search results transfers RQS history search results from the output interface of RQSM History Maintenance to the input interface of RQS Modification.
- The private link RQS assessment history search results transfers RQS assessment history search results from the output interface of RQSM History Maintenance to the input interface of RQS Modification.
- The private link DOD assessment history search results transfers DOD assessment history search results from the output interface of RQSM History Maintenance to the input interface of RQS Modification.
- The private link current RQS replacement results transfers current RQS replacement results from the output interface of RQS Modification to the input interface of RQSM History Maintenance.
- The private link new current RQS transfers new current RQS contents from the output interface of RQSM History Maintenance to assumptions on design requirement information in the input interface of current RQS maintenance on the basis of an explicit mapping between these information types.
- The private link current RQS to be stored transfers epistemic information on design requirement information from the output interface of current RQS maintenance to current RQS contents in the input interface of RQSM History Maintenance on the basis of an explicit mapping between these information types.
- The private link current RQS to be analysed transfers epistemic information on design requirement information from the output interface of current RQS maintenance to current RQS contents in the input interface of RQS Modification.
- The private link basic design requirement information transfers basic design requirement information from the output interface of current RQS maintenance to the input interface of deductive RQS refinement.
- The private link design requirement information transfers design requirement information from the output interface of deductive RQS refinement to the input interface of current RQS maintenance.
- The mediating link RQS transfers RQS from the output interface of RQSM History Maintenance to the output interface of RQS Manipulation.

- The mediating link RQS assessment transfers RQS assessment from the output interface of RQS Modification to the output interface of RQS Manipulation.
- The mediating link RQSM process evaluation transfers RQSM process evaluation from the output interface of RQS Modification to the output interface of RQS Manipulation.

### A.3.3 Knowledge composition related to RQS manipulation

Additional information types are described, which are related to the information types RQS modification progress, RQS modification state history, and RQS history.

**RQS modification progress.** The information type RQS modification progress, see Figure A.9, describes the current state of the modification process, and the information available on success and/or failure of the modification process. This information type consists of current RQS modification state, and RQSM process evaluation. The information type RQSM process evaluation is described in Section A.1. The information type current manipulation action is related to the information type RQS modification progress: manipulation actions influence the sub-processes within RQS manipulation.

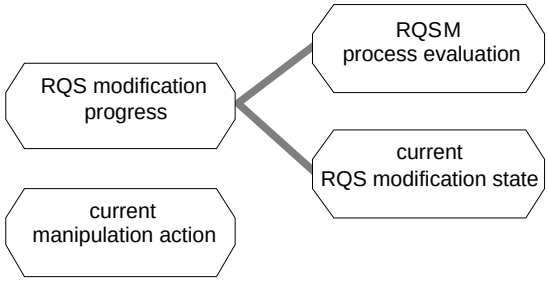


**Figure A.9** Information types related to RQS modification progress.

The information type current manipulation action was described in the previous section. The generic relation defined in the information type current RQS modification state is:

**relations**
is_current_RQS_modification_state_value:
RQS_modification_state_attribute *
RQS_modification_state_attribute_value;

This relation describes the contents of the current modification state, without attaching an identifier (i.e., the sort RQS modification state). Within the RQSM History Maintenance an identifier is attached, as well as additional temporal and persistent information (e.g., the final RQS).

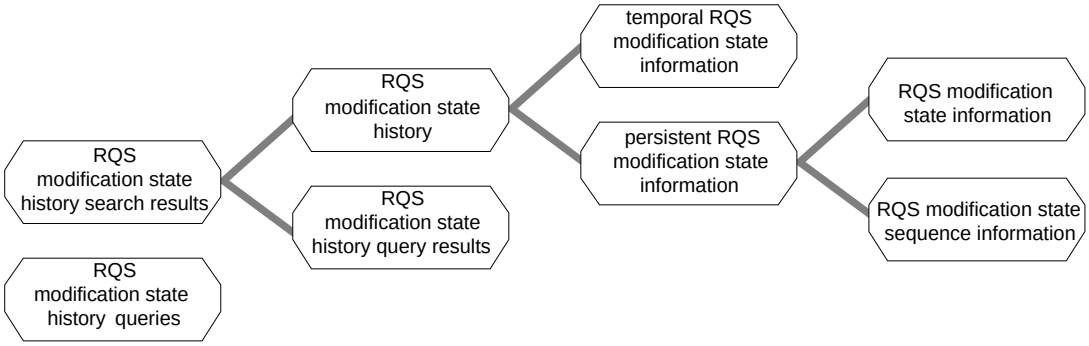


**Figure A.10** Information types related to RQS modification state history.

**RQS modification state history.** The information type RQS modification state history is employed to describe information related to modification ‘steps’. It is related to a number of

information types, as shown in Figure A.10. The information type RQS modification state history consists of two information types: temporal RQS modification state information, and persistent RQS modification state information. The latter information type consists of two information types: RQS modification state information, and RQS modification state sequence information. Three other information types are conceptually related to RQS modification state history: RQS modification state history queries, RQS modification state history search results, and RQS modification state history query results.

Generic relations defined in the information type temporal RQS modification state information are:

**relations**
is_newest_RQS_modification_state,
is_initial_RQS_modification_state: RQS_modification_state;

These relations denote which RQS modification state is the latest, newest (which is relative to modification state steps taken by RQS Modification), and which modification state is the initial modification state (which is relative to each 'design problem setting').

The generic relation defined in the information type RQS modification state information is:

**relations**
has_RQS_modification_state_value: RQS_modification_state *
RQS_modification_state_attribute *
RQS_modification_state_attribute_value;

This relation can be used to describe which overall design strategy is related to a modification state, which modifications are related to a modification state, on which RQS the modifications are based, which RQS is the result of this modification step, etc.

Generic relations defined in the information type RQS modification state sequence information are:

**relations**
has_preceding_RQS_modification_state,
has_succeeding_RQS_modification_state: RQS_modification_state *
RQS_modification_state;
is_first_RQS_modification_state: RQS_modification_state;

These relations chain together modification states, and define which modification state was the very first modification state in the RQSM history.

The information types RQS modification state history queries and RQS modification state history query results contain relations with which queries can be formulated on the modification state history, and relations with which the results of the queries can be expressed.

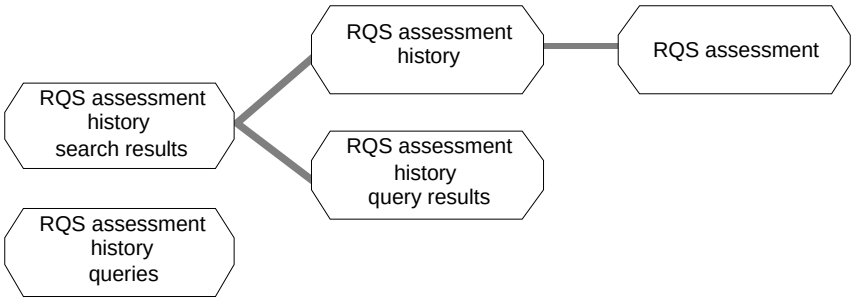


**Figure A.11** Information types related to RQS assessment history.

**RQS assessment history.** The information type RQS assessment history is employed to retain information on assessments of sets of qualified requirements (comparisons among sets of qualified requirements, etc.). It is related to a number of information types, as shown in Figure A.11. The information type RQS assessment history consists only of the information type RQS assessment. Three other information types are conceptually related to RQS assessment history: RQS assessment history queries, RQS assessment history search results, and RQS assessment

history query results. The information type RQS assessment history search results refers to RQS assessment history and RQS assessment history query results.

The information type RQS assessment is described in Section 6.1.2. The information types RQS assessment history queries and RQS assessment history query results contain relations with which queries can be formulated on the RQS assessment history, and relations with which the results of the queries can be expressed.

**RQS history.** The information type RQS history is employed to retain information on requirement qualification sets. This information type is related to a number of information type, as shown in Figure A.12. The information type RQS history consists of two information types: temporal RQS information, and persistent RQS information The latter information type consists of two information types: RQS and RQS sequence information. Three information types related to the information type RQS history are RQS history queries, RQS history search results, and RQS history query results. The information type RQS history search results refers to RQS history and RQS history query results.

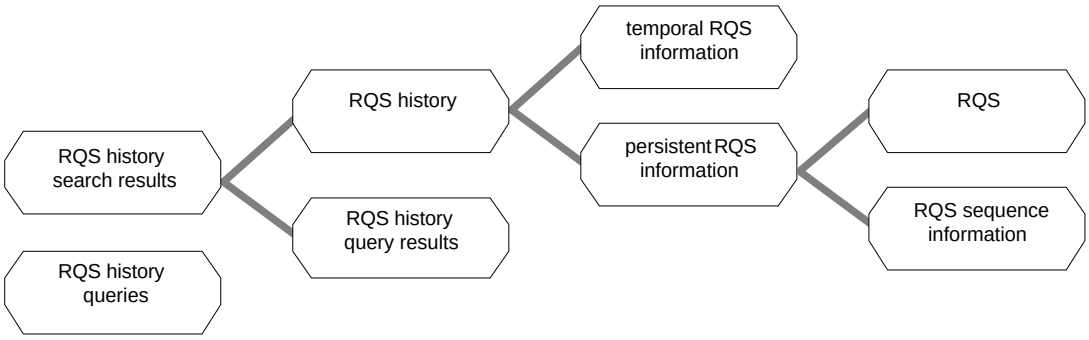


**Figure A.12** Information types related to RQS history.

The information type RQS is described in Section 6.1. The generic relation defined in the information type temporal RQS information is:

**relations**
is_newest_RQS,
is_initial_RQS: RQS_name;

These relations denote which RQS is the latest, newest (i.e., last stored via a current RQS contents, or as an initial RQS), and which RQS is the initial RQS (which is relative to each 'design problem setting').

The generic relations define in the information type RQS sequence information are:

**relations**
has_previous_RQS,
has_next_RQS: DOD_name * DOD_name;
is_first_RQS: DOD_name;

These relations chain together sets of qualified requirements, and define which set of qualified requirements was the very first set in the RQSM history.

The information types RQS history queries, and RQS history query results contain relations with which queries can be formulated on the RQS history, and relations with which the results of the queries can be expressed.

# B Additional Descriptions of the Re-design Model for Compositional Systems

The refinement of the generic model of design is described in Chapters 7, 8 and 9. A number of refinements of the process composition and knowledge composition related to the refinement of the generic model of design are not described in Chapters 7, 8 and 9; these refinements are described in this appendix.

Section B.1 describes additional refinements for RQS manipulation. Section B.2 describes additional refinements for DOD manipulation.

## B.1 Additional refinements for RQSM

The refinement of several sub-components of RQS manipulation has not been described in Chapter 8. The component RQS modification process co-ordination was introduced in Section 8.3.2 as a sub-component of RQS modification and its composition is described in Section B.1.1. The component default extension method was introduced in Section 8.3.5 as a sub-component of RQS modification determination and its composition is described in Section B.1.2. The sub-components of RQSM history maintenance have been introduced in Section 8.4.2; their compositions are described in Section B.1.3. Additional knowledge structures related to RQSM are described at the end of each of these three sections.

### B.1.1 Refinement of RQS modification process co-ordination

The composition of RQS modification process co-ordination is described by process composition and knowledge composition.

**Process composition for RQS modification process co-ordination: identification of processes and abstraction levels.** The process composition of RQS modification process co-ordination is described by levels of process abstraction, identification of processes, and composition relation between processes.

The process RQS modification process co-ordination is a sub-process of RQS modification, as described in Section 8.3.2. The first level of process abstraction for RQS modification process co-ordination is shown in Figure B.1. The processes RQS modification progress co-ordination, RQSM history navigation, RQS modification state analysis, and RQS modification strategy determination are distinguished with the process RQS modification process co-ordination.

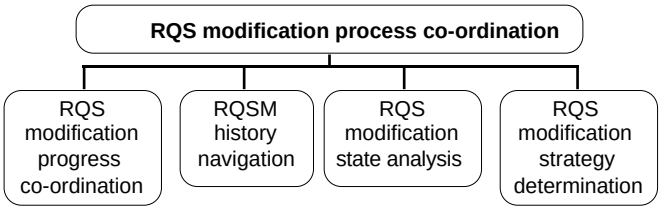


**Figure B.1.** Partial process refinement for RQS modification process co-ordination.

The process RQS modification progress co-ordination is responsible for strategic control within the entire RQSM process on the basis of current information on the four sub-processes of RQSM. This includes deciding e.g. when to navigate the history, when to validate a specific RQS, etc.

The sub-process RQSM history navigation is responsible for querying the history component (on RQS modification states, and sets of qualified requirements) to find the best suited RQS to continue the modification process, on the basis of available information on global design strategies and manipulation actions. The process RQS modification state analysis analyses a state of the modification process. The process RQS modification strategy determination determines, on the basis of analyses of modification states, which modification strategy is to be employed.

Each of the processes depicted in Figure B.1 can be characterised in terms of their input and output information types, these information types are listed in Table B.1 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| RQS modification progress co-ordination | • overall design strategy<br>• history navigation results<br>• RQS modification strategy<br>• current modification state contents | • current manipulation action<br>• RQS modification progress<br>• history navigation directives<br>• RQS assessment<br>• RQS modification strategy preferences |
| RQSM history navigation | • history navigation directives<br>• RQS modification state history search results<br>• RQS assessment history search results<br>• DOD assessment history search results<br>• RQS history search results<br>• current RQS replacement results<br>• temporal modification state contents analysis | • history navigation results<br>• RQS modification state history queries<br>• RQS assessment history queries<br>• DOD assessment history queries<br>• RQS history queries<br>• current RQS replacement request<br>• historical modification state content |
| RQS modification state analysis | • modification state content | • modification state contents analysis |
| RQS modification strategy determination | • RQS modification strategy preferences<br>• current modification state analysis<br>• historical modification state analysis | • RQS modification strategy |

**Table B.1** Interface information types for sub-processes of RQS modification process co-ordination.

- The process RQS modification progress co-ordination requires the overall design strategy (overall design strategy), navigation results (history navigation results), RQS modification strategy (RQS modification strategy), and the contents of the current modification state (current modification state contents). This process generates manipulation actions for the overall RQSM process (current manipulation action), information on the progress of the modification process (RQS modification progress), directives for history navigation (history navigation directives), assessments of sets of qualified requirements (RQS assessment), and preferences on modification strategies (RQS modification strategy preferences).
- The process RQSM history navigation needs directives for navigation (history navigation directives), results of searching the RQS modification state history ( RQS modification state history search results), results of searching the RQS assessment history (RQS assessment history search results), results of searching the DOD assessment history (DOD assessment history search results), results of searching the RQS history (RQS history search results), results on success of replacing the current RQS (current RQS replacement results), and analysis of the contents of a number of modification states (temporal modification state contents analysis). This process produces navigation results (history navigation results), queries on RQS modification state history ( RQS modification state history queries, queries on RQS assessment history (RQS assessment history queries), queries on DOD assessment history (DOD assessment history queries), queries on RQS history (RQS history queries),

requests for replacement of the current RQS (current RQS replacement request), and contents of a (historical) modification state (historical modification state contents).
- The process RQS modification state analysis needs the contents of a modification state (modification state content, i.e., modification foci, current modifications, and assessment of the current RQS). This process generates an analysis of the given modification state contents (modification state contents analysis).
- The process RQS modification strategy determination requires preferences on RQS modification strategies (RQS modification strategy preferences), analysis of the current modification state (modification state analysis), and analysis of a previous RQS modification state (modification state analysis). This process produces a strategy for the modification of the current RQS (RQS modification strategy).

**Process composition relations within RQS modification process co-ordination.** The information links in the component RQS modification process co-ordination are shown in Figure B.2.

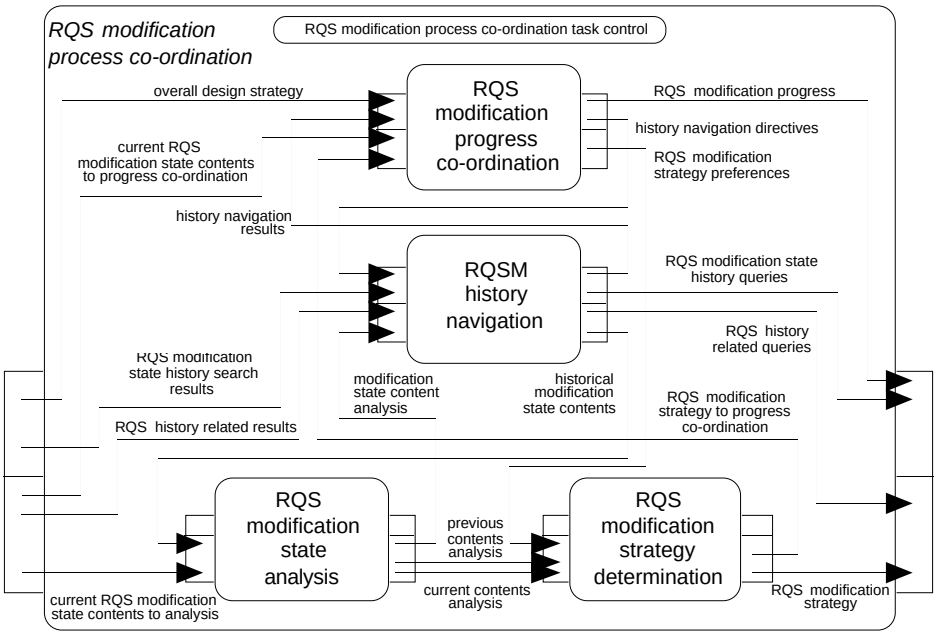


**Figure B.2** Information links within the process of RQS modification process co-ordination.

Within this component nine mediating links and seven private links are defined:

- The mediating links current RQS modification state contents to analysis and current RQS modification state contents to progress co-ordination transfer information expressed in modification state contents from the input interface of RQS modification process co-ordination to the input interfaces of RQS modification state analysis and RQS modification progress co-ordination, respectively.
- The mediating link overall design strategy transfers information expressed in overall design strategy from the input interface of RQS modification process co-ordination to the input interface of RQS modification progress co-ordination.
- The mediating links RQS modification state history search results, and RQS history related results transfer information expressed in RQS modification state history search results, RQS assessment history search results, DOD assessment history search results, current RQS replacement results, and RQS history search results from the input interface of RQS modification process co-ordination to the input interface of RQSM history navigation.
- The private links history navigation directives and history navigation results transfer information expressed in history navigation directives and history navigation results, respectively, between RQS modification progress co-ordination and RQSM history navigation.

- The private links historical modification state contents and modification state contents analysis transfer information expressed in historical modification state contents and modification state contents analysis, respectively, between RQSM history navigation and RQS modification state analysis.
- The private links RQS modification strategy preferences and RQS modification strategy to progress co-ordination transfer information expressed in RQS modification strategy preferences and RQS modification strategy, respectively, between RQS modification progress co-ordination and RQS modification strategy determination.
- The private links previous contents analysis and current contents analysis transfer information expressed in modification state content analysis to historical modification state analysis and current modification state analysis, respectively, from the output interface of RQS modification state analysis to the input interface of RQS modification strategy determination, on the basis of an explicit mapping between these information types.
- The mediating link RQS modification strategy transfers information expressed in RQS modification strategy from the output interface of RQS modification strategy determination to the output interface of RQS modification process co-ordination.
- The mediating links RQS modification state history queries and RQS history related information transfer information related to the RQSM history from the output interface of RQSM history navigation to the output interface of RQS modification process co-ordination.
- The mediating link RQS modification progress transfers information expressed in RQS modification progress from the output interface of RQS modification progress co-ordination to the output interface of RQS modification process co-ordination.

The task control within the component RQS modification process co-ordination is as follows. Upon activation of RQS modification process co-ordination, RQS modification progress co-ordination is activated which co-ordinates the sub-processes within RQS modification process co-ordination. A distinction can be made between continuation of the previous manipulation process, or initiation of a new manipulation process. The global phases within RQS modification process co-ordination resemble a process control model. In a process control task a cycle occurs over the sub-tasks: analysis, planning, and execution. Similarly within RQS modification process co-ordination analysis is performed by RQS modification state analysis and RQSM history navigation, planning is performed by RQS modification strategy determination and RQS modification progress co-ordination, and execution is performed by effectuating modification strategies, resulting in modifications to the current RQS.

RQSM history navigation is able to formulate queries on histories, requests for replacement of the current RQS, and isolate contents of a historical modification state. RQS modification state analysis can be used to analyse the contents of the current modification state, or the contents of a previous modification state (by means of the links current RQS modification state contents and historical modification state contents). RQS modification strategy determination can be activated to determine a modification strategy.

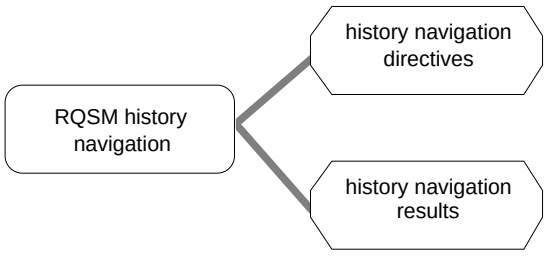


**Figure B.3** Information types related to RQSM history navigation.

**Knowledge composition for RQS modification process co-ordination.** The information types history navigation directives, history navigation results, modification strategy preferences, RQS modification state contents, historical RQS modification state contents,

modification state contents analysis, previous modification state contents analysis, and current modification state contents analysis are described here. Two knowledge bases are described at the end if this section.

Two of the information types related to the process RQSM history navigation are shown in Figure B.3. The information type history navigation directives contains directives ('goals') for the RQSM navigation process. The information type history navigation results contains information on the achievement of the given navigation directives.

Relations in the information type history navigation directives are:

**relations**

| | |
|---|---|
| RQS_to_become_current: | RQS_name; |
| RQS_contents_to_be_retrieved: | RQS_name; |
| comparable_RQS_to_be_retrieved_for: | comparison_type * |
| | RQS_name; |

The relation RQS to become current specifies that the named RQS is to become the current RQS: the contents of the named RQS are to be placed in current RQS maintenance. The relation RQS contents to be retrieved describes which specific RQS has to be retrieved from the RQSM history. The relation comparable RQS to be retrieved for specifies that a RQS has to be retrieved which is comparable with the named RQS, e.g., a refinement of the named RQS.

Relations in the information type history navigation results are:

**relations**

| | |
|---|---|
| RQS_contents_retrieved_for: | RQS_name; |
| comparable_RQS_for: | RQS_name * |
| | comparison_type * |
| | RQS_name; |

The relation RQS contents retrieved for specifies that the contents of the named RQS have been retrieved. The relation comparable RQS for specifies that a comparable RQS has been found.

The information type RQS modification strategy preferences is shown in Figure B.4. It contains information on which direction of modification is to be preferred, including, e.g., previously failed modification strategies. Both RQS modification process co-ordination and RQS modification strategy determination make use of this information type.

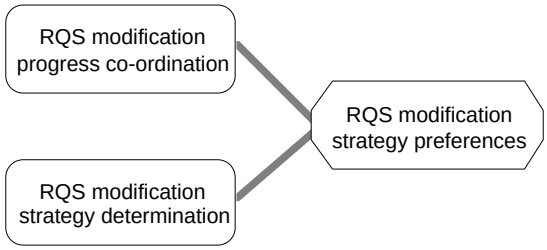


**Figure B.4** The information type RQS modification strategy preferences.

A relation in the information type RQS modification strategy preferences is:

**relations**

| | |
|---|---|
| rejected_modification_strategy: | modification_method_identification * |
| | modification_method_characterisation; |

The relation rejected modification strategy specifies which modification strategy is not to be used.

Two information types related to the contents of a modification state are shown in Figure B.5. The information type RQS modification state contents describes current RQS basis evaluation, modification foci, and modifications. The information type historical RQS modification state contents describes the same information. Both information types refer to the same information types, yet their usage in components requires the distinction between contents of *a* modification state, and the contents of a *historical* modification state.

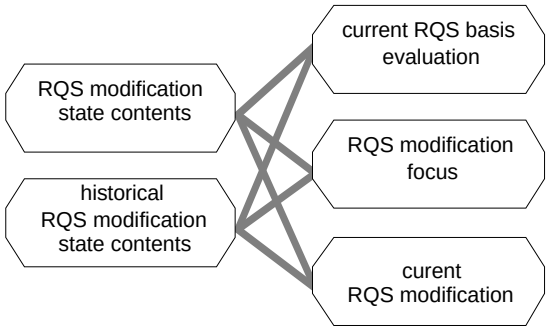


**Figure B.5** Information types related to the contents of a modification state.

The information types related to the analysis of the contents of a modification state are shown in Figure B.6. The information type modification state contents analysis describes the analysis of *a* modification state. The information type temporal modification state contents analysis contains descriptions of an analysis of the contents of a previous and a current modification state. The information type previous modification state contents analysis describes the analysis of a *previous* modification state contents, and the information type current modification state contents analysis describes the analysis of the *current* modification state contents. The process RQS modification state analysis makes use of the information type modification state contents analysis, and the process RQSM history navigation makes use of the information type temporal modification state contents analysis.

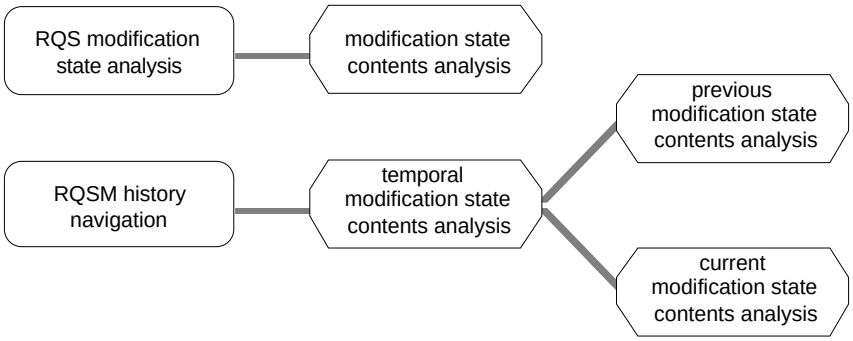


**Figure B.6** Information types related to analysis of the contents of a modification state.

The relation in the information type modification state contents analysis is:

**relations**
content_analysis: RQS_modification_state_content_analysis;

The relation content analysis specifies whether the contents of a modification state contains, e.g., conflicts among requirements, requirements which have been assessed in relation to a DOD, and requirements which are not refinements and are not refined at all.

The relations in the information types previous modification state contents analysis and current modification state contents analysis are:

**relations**
previous_content_analysis: RQS_modification_state_content_analysis;
current_content_analysis: RQS_modification_state_content_analysis;

These two relations specify the analysis of either the contents of a previous modification state, or the current modification state.

Two knowledge bases, related to sub-processes of RQS modification process co-ordination, are shown in Figure B.7.

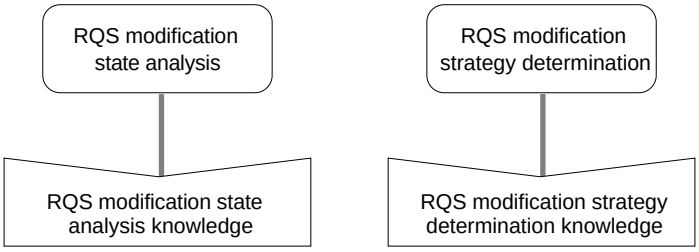


**Figure B.7** Two knowledge bases related to sub-processes of RQS modification process co-ordination.

The knowledge base RQS modification state analysis knowledge is employed to analyse the contents of a modification state. An example from this knowledge base is shown below.

**Example from knowledge base RQS modification state analysis knowledge**

The knowledge element below illustrates that a qualified requirement which does not have a refinement, nor is a refinement of another qualified requirement leads to the conclusion that a non-refined, non-is-a-refinement qualified requirement exists in the current RQS.

```
if   validation_result( has_refinement( QuRe: qualified_requirement_name ), neg )
     and   validation_result( is_a_refinement( QuRe: qualified_requirement_name ), neg )
then      content_analysis( contains_non_refined_non_is_a_refinement_qr );
```

The knowledge base RQS modification strategy determination knowledge is employed to determine strategies for the modification process. It is important to note that whenever a RQS is made the current RQS, it is always validated, and after validation a modification strategy is determined (on the basis of the validation results). An example is shown below.

**Example from knowledge base RQS modification strategy determination knowledge**

The knowledge element below illustrates that if it is known that this is the first stage of design, and it is known that conflicting design requirements are present, then the modification strategy is to resolve these conflicts.

```
if   initial_stage_of_RQSM
     and   content_analysis( contains_apparent_conflicts )
then      RQS_modification_strategy( resolve, apparent_conflicts );
```

### B.1.2 Refinement of default extension method

The composition of default extension method is described by process composition and knowledge composition.

**Process composition for default extension method: identification of processes and abstraction levels.** The process composition of default extension method is described by levels of process abstraction, identification of processes, and composition relation between processes.

The process default extension method is a sub-process of RQS modification determination, as described in Section 8.3.5. The first level of process abstraction for default extension method is shown in Figure B.8. The processes default extension alternative determination and default extension determination are distinguished within the process default extension method.

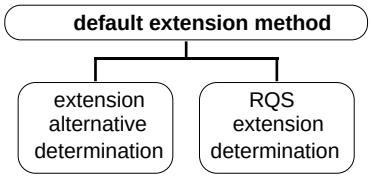


**Figure B.8** Process composition of default extension method.

The default extension method for sets of qualified requirements entails two processes: default extension alternative determination, which determines which extension alternative is to be chosen, and the process default extension determination, in which design requirements in focus are extended. Currently the extension knowledge is based on refinements of properties on compositional systems, as described in Chapter 5.

| *process* | *input information type* | *output information type* |
|---|---|---|
| default extension alternative determination | • modification method<br>• RQS modification focus | • selected extension alternative |
| default extension determination | • RQS modification focus<br>• current RQS contents<br>• rejected RQS modification<br>• selected extension alternative | • extension results |

**Table B.2** Interface information types for sub-processes of default extension method.

The interface information types of the sub-processes of default extension method are listed in Table B.2 and described below.

- The process default extension alternative determination requires a modification method (modification method), and a focus for modification (RQS modification focus). This process produces a selected extension alternative (selected extension alternative).
- The process default extension determination needs a focus for modification (RQS modification focus), the contents of the current RQS (current RQS contents), rejected modifications (rejected RQS modification), and a selected extension alternative (selected extension alternative). As its results it generates extensions on the current RQS (extension results).

**Process composition relations within default extension method.** The information links in the component default extension method are shown in Figure B.9.

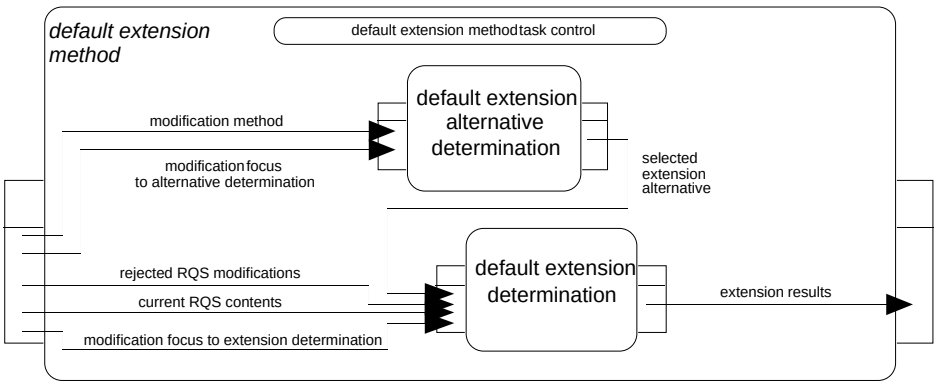


**Figure B.9** Information links within the process of default extension method.

Within this component six mediating links and one private link are defined:

- The mediating link modification method transfers information on the modification method (modification method) from the input interface of default extension method to the input interface of default extension alternative determination.
- The mediating link modification focus to alternative determination transfers foci for modification (selected modification focus) from the input interface of default extension method to the input interface of default extension alternative determination.
- The mediating link rejected RQS modifications transfers rejected modifications (rejected RQS modification) from the input interface of default extension method to the input interface of default extension determination.

- The mediating link current RQS contents transfers the contents of the current RQS (current RQS contents) from the input interface of default extension method to the input interface of default extension determination.
- The mediating link modification focus to extension determination transfers foci for modification (selected modification focus) from the input interface of default extension method to the input interface of default extension determination.
- The private link selected extension alternative transfers the extension alternative to consider (selected extension alternative) from the output interface of default extension alternative determination to the input interface of default extension determination.
- The mediating link extension results transfers extensions for a RQS (extension results) from the output interface of default extension determination to the output interface of default extension method.

The task control within the component default extension method is as follows. Upon activation of default extension method the information links modification method, and modification focus to alternative determination are made up-to-date and default extension alternative determination is activated. Upon termination of default extension alternative determination, default extension determination is activated and the information links selected extension alternative, rejected modifications, current RQS contents, and modification focus to extension determination are made up-to-date. Upon termination of default extension determination the information link extension results is made up-to-date and default extension method terminates itself.

**Knowledge refinement for default extension method.** One information type related to default extension method has not been described in Section 8.3.6 and is shown in Figure B.10. The information type selected extension alternative contains information on which extension alternative is to be employed when extending the design requirement currently in focus. This information type refers to the information type selected refinement alternative: the process default extension method employs knowledge on possible refinements for design requirements.

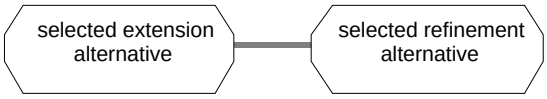


**Figure B.10** Information type selected extension alternative.

The relation defined in the information type selected refinement alternative is:

```
relations
    selected_refinement_alternative:        refinement_alternative;
```

The relation selected refinement alternative specifies which refinement alternative (i.e., specialisation or realisation) is relevant (with respect to the current modification focus).

The knowledge base related to the process default extension determination consists of the knowledge base extension by refinement knowledge, as shown in Figure B.11.

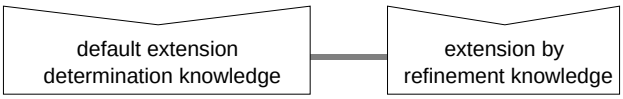


**Figure B.11** Knowledge base default extension determination knowledge.

The knowledge base extension by refinement knowledge contains instances of knowledge such as the following two knowledge elements. These two knowledge elements illustrate the refinement of a qualified requirements in focus: both refinement alternatives are shown.

```
if  is_qualified_requirement_selected_as_focus( QR: qualified_requirement_name )
    and   holds( is_qualified_requirement(  QR: qualified_requirement_name,
                                            Q: qualification,
                                            R: requirement_name ),
                 pos )
    and   holds( is_requirement(            R: requirement_name,
                                            has_property( A: agent_name,
                                                  is_capable_of_bidirectional_communication_with(
                                                        A2: agent_name )) ),
                 pos )
    and   selected_refinement_alternative( specialisations )
then      addition_to_current_RQS(
              is_qualified_requirement(     new_name( QR: qualified_requirement_name, a ),
                                            Q: qualification,
                                            new_name( R: requirement_name, a ) ) )
    and   addition_to_current_RQS(
              is_requirement(               new_name( R: requirement_name, a ),
                                            has_property( A: agent_name,
                                                  is_capable_of_unidirectional_communication_from(
                                                        A2: agent_name ) ) )
    and   addition_to_current_RQS(
              is_qualified_requirement(     new_name( QR : qualified_requirement_name, b ),
                                            Q: qualification,
                                            new_name( R: requirement_name, b ) ) )
    and   addition_to_current_RQS(
              is_requirement(       new_name( R: requirement_name, b ),
                                            has_property( A: agent_name,
                                                  is_capable_of_unidirectional_communication_to(
                                                        A2: agent_name ) ) )
    and   addition_to_current_RQS(
              is_qualified_requirement(     new_name( QR: qualified_requirement_name, c ),
                                            Q: qualification,
                                            new_name( R: requirement_name, c ) ) )
    and   addition_to_current_RQS(
              is_requirement(               new_name( R: requirement_name, c ),
                                            has_property( A: agent_name,
                                            is_capable_of_combining_unidirectional_communication_
                                                  from_and_to( A2: agent_name ) ) );

if  is_qualified_requirement_selected_as_focus( QR: qualified_requirement_name )
    and   holds( is_qualified_requirement(  QR: qualified_requirement_name,
                                            Q: qualification,
                                            R: requirement_name ),
                 pos )
    and   holds( is_requirement(            R: requirement_name,
                                            has_property( A: agent_name,
                                                  is_capable_of_unidirectional_communication_from(
                                                        A2: agent_name )) ),
                 pos )
    and   selected_refinement_alternative(realisations )
then      addition_to_current_RQS(
              is_qualified_requirement(     new_name( QR: qualified_requirement_name, a ),
                                            Q: qualification,
                                            new_name( R: requirement_name, a ) ) )
    and   addition_to_current_RQS(
              is_requirement(               new_name( R: requirement_name, a ),
                                            has_property( A: agent_name,
                                                  is_capable_of_reasoning_about_unidirectional_
                                                  communication_from( A2: agent_name ) ) )
```

```
and   addition_to_current_RQS(
          is_qualified_requirement(   new_name( QR : qualified_requirement_name, b ),
                                      Q: qualification,
                                      new_name( R: requirement_name, b ) ) )
and   addition_to_current_RQS(
          is_requirement(             new_name( R: requirement_name, b ),
                                      has_property( A: agent_name,
                                            is_capable_of_executing_unidirectional_
                                            communication_to( A2: agent_name ) ) )
and   addition_to_current_RQS(
          is_qualified_requirement(   new_name( QR: qualified_requirement_name, c ),
                                      Q: qualification,
                                      new_name( R: requirement_name, c ) ) )
and   addition_to_current_RQS(
          is_requirement(             new_name( R: requirement_name, c ),
                                      has_property( A: agent_name,
                                      is_capable_of_combining_reasoning_and_executing_
                                      unidirectional_communication_from( A2: agent_name ) ) );
```

### B.1.3 Additional refinement of RQSM history maintenance

The composition of RQS history maintenance and RQS modification state history maintenance, sub-processes of RQSM history maintenance is described by process composition and knowledge composition.

**Process composition for sub-processes of RQSM history maintenance: identification of processes and abstraction levels.** The process compositions for the three sub-processes of RQSM history maintenance is described by levels of process abstraction, identification of processes, and composition relation between processes.

The process RQSM history maintenance is described in Section 8.4.2. The first level of process abstraction for sub-processes of RQSM history maintenance is shown in Figure B.12. The processes RQS storage preparation, RQS permanent storage, and RQS retrieval are distinguished with the process RQS history maintenance. The processes RQS modification state storage preparation, RQS modification state permanent storage, and RQS modification state retrieval are distinguished with the process RQS modification state history maintenance.

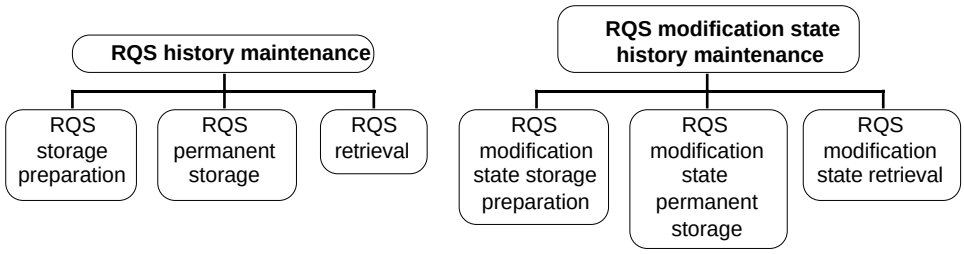


**Figure B.12** Processes at different abstraction levels for RQSM history maintenance.

The process RQS history maintenance is responsible for storing, retrieving and managing sets of design requirements. This process is composed of three sub-processes: the process RQS storage preparation assigns names to sets of design requirements and relates the name of a set to specific contents of the set. The process RQS permanent storage stores the result of the previous process permanently, available for later retrieval. The process RQS retrieval retrieves (parts of) sets of design requirements on the basis of retrieval queries.

The process RQS modification state history maintenance is responsible for storing, retrieving and managing modification states (i.e. information regarding the modification process). This process is similar to the process RQS history maintenance. The process RQS modification state history maintenance is composed of three sub-processes. The process RQS modification state storage preparation is responsible for preparing information to be stored, this entails determining a unique name for the new modification state and relating information on the

modification state with that unique name. The process RQS modification state permanent storage is responsible for storing the resulting RQS modification state and making it available for the retrieval of (parts of) modification states by the process RQS modification state retrieval, which guides its retrieval on the basis of specific requests for specific information.

The interface information types of the sub-processes of RQS history maintenance and RQS modification state history maintenance are listed in Table B.3 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| *Sub-processes of RQS history maintenance* | | |
| RQS storage preparation | • RQS<br>• current RQS contents | • persistent RQS information to be stored<br>• temporal RQS information<br>• current RQS name |
| RQS permanent storage | • persistent RQS information to be stored<br>• DOD assessment<br>• RQS assessment | • persistent RQS information<br>• DOD assessment<br>• RQS assessment |
| RQS retrieval | • persistent RQS information<br>• RQS history queries<br>• current RQS replacement request<br>• DOD assessment<br>• DOD assessment history queries<br>• RQS assessment<br>• RQS assessment history queries | • RQS history query results<br>• new current RQS contents<br>• persistent RQS information<br>• current RQS replacement results<br>• DOD assessment search results<br>• RQS assessment search results |
| *Sub-processes of RQS modification state history maintenance* | | |
| RQS modification state storage preparation | • given current RQS name<br>• current RQS modification state contents | • persistent RQS modification state information to be stored<br>• temporal RQS modification state information |
| RQS modification state permanent storage | • persistent RQS modification state to be stored | • persistent RQS modification state information |
| RQS modification state retrieval | • RQS modification history queries<br>• persistent RQS modification state information | • RQS modification history query results<br>• persistent RQS modification state information |

**Table B.3** Interface information types for sub-processes of RQS history maintenance.

The input and output information in the interface of the sub-processes of RQS history maintenance is described below:

- The process RQS storage preparation requires information on descriptions of design objects (RQS) and the contents of the current RQS (current RQS contents). This process produces a name for the current RQS (current RQS name), temporal information on sets of qualified requirements (temporal RQS information), and persistent information on sets of qualified requirements to be stored (persistent RQS information to be stored).
- The process RQS permanent storage needs information on persistent information on sets of qualified requirements to be stored (persistent RQS information to be stored), assessments of design object descriptions (DOD assessment), and assessments of sets of qualified requirements (RQS assessment). This process generates persistent information on sets of qualified requirements to be stored (persistent RQS information), assessments of design object descriptions (DOD assessment), and assessments of sets of qualified requirements (RQS assessment).
- The process RQS retrieval requires information on persistent information on sets of qualified requirements (persistent RQS information), queries on RQS history (RQS history

queries), and requests for replacement of the current RQS (current RQS replacement request), assessments of design object descriptions (DOD assessment), queries on DOD assessment history (DOD assessment history queries), assessments of sets of qualified requirements (RQS assessment), and queries on RQS assessment history (RQS assessment history queries). This process produces results of searching the DOD assessment history (DOD assessment history search results), results of searching the RQS assessment history (RQS assessment history search results), results of queries on RQS history (RQS history query results), persistent information on sets of qualified requirements (persistent RQS information), new contents for current RQS maintenance (new current RQS contents), and results on the success of replacing the current RQS (current RQS replacement results).

The input and output information in the interface of the sub-processes of RQS modification state history maintenance is described below:

- The process RQS modification state storage preparation requires information on the name given to the current RQS (given current RQS name), contents of the current modification state (current RQS modification state contents). This process produces temporal information on RQS modification states (temporal RQS modification state information), and persistent information on RQS modification states to be stored (persistent RQS modification state information to be stored).
- The process RQS modification state permanent storage needs information on persistent information on RQS modification states to be stored (persistent RQS modification state information to be stored). This process generates persistent information on RQS modification states (persistent RQS modification state information).
- The process RQS modification state retrieval requires information on persistent information on RQS modification states (persistent RQS modification state information), and queries on RQS modification state history (RQS modification state history queries). This process produces results of queries on RQS modification state history (RQS modification state history query results), and persistent information on RQS modification states (persistent RQS modification state information).

**Process composition relations within RQS history maintenance.** The information links in the component RQS history maintenance are shown in Figure B.13.

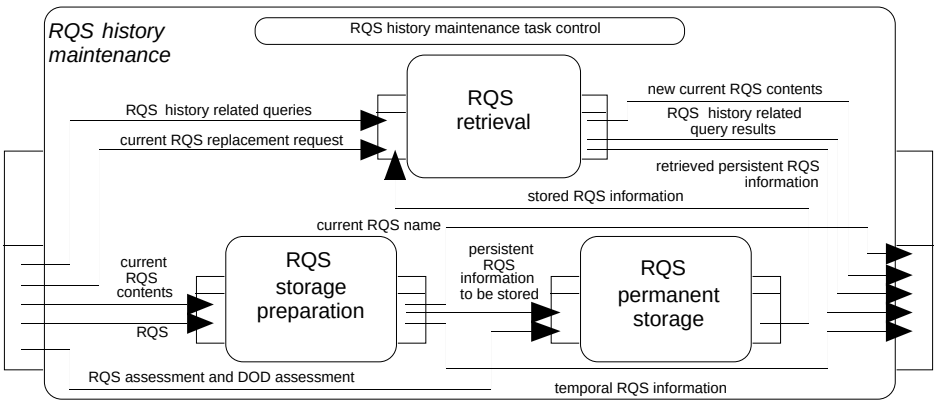


**Figure B.13** Information links within the process of RQS history maintenance.

Within this component ten mediating links and two private links are defined:

- The mediating links RQS and current RQS contents transfer sets of qualified requirements (RQS) and the contents of the current RQS (current RQS contents), respectively, from the input interface of RQS history maintenance to the input interface of RQS storage preparation.

- The mediating link RQS assessment and DOD assessment transfers assessments of design object descriptions (DOD assessment) and assessments of sets of qualified requirements (RQS assessment) from the input interface of RQS history maintenance to the input interface of RQS permanent storage.
- The information links RQS history related queries and current RQS replacement request transfer queries on RQS history (RQS history query specification), queries on DOD assessment history (DOD assessment history queries), and queries on RQS assessment history (RQS assessment history queries); and requests for replacement of the current RQS (current RQS replacement request), respectively, from the input interface of RQS history maintenance to the input interface of RQS retrieval.
- The private link persistent RQS information to be stored transfers information expressed in the information types with the same name from the output interface of RQS storage preparation to the input interface of RQS permanent storage.
- The private link stored RQS information transfers information expressed in persistent RQS information, DOD assessment, and RQS assessment from the output interface of RQS permanent storage to the input interface of RQS retrieval.
- The mediating links current RQS name and temporal RQS information transfer the name for the current RQS (current RQS name) and temporal information on sets of qualified requirements (temporal RQS information), respectively, from the output interface of RQS storage preparation to the output interface of RQS history maintenance.
- The mediating links new current RQS contents, RQS history related query results, and retrieved persistent RQS information transfer the new contents of the current RQS (new current RQS contents), results of queries on histories (RQS history query results, DOD assessment history search results, and RQS assessment history search results), and persistent information on design object descriptions (persistent RQS information), respectively, from the output interface of RQS retrieval to the output interface RQS history maintenance.

The task control within the component RQS history maintenance is as follows. Upon activation of RQS history maintenance a number of situations are possible, which correspond to task control foci for this component. The task control foci are: initial storage of information, continued storage of information, update of the history, execute queries, and replacement of current RQS preparation. Depending on the specific task control focus, specific links and sub-components are activated, e.g., for replacement of current RQS preparation, the component RQS retrieval is activated and the information link current RQS replacement request is made up-to-date. Upon termination of RQS retrieval, the information links new current RQS contents and RQS history related query results are made up-to-date and RQS history maintenance is terminated.

**Process composition relations within RQS modification state history maintenance.** The information links in the component RQS modification state history maintenance are shown in Figure B.14.

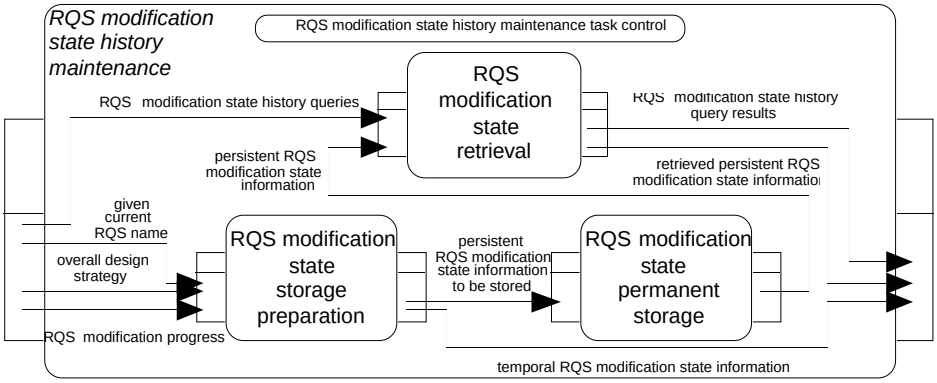


**Figure B.14** Information links within the process of RQS modification state history maintenance.

Within this component seven mediating links and two private links are defined:

- The mediating links given current RQS name, overall design strategy, and RQS modification progress transfer the name of the current RQS (given current RQS name), the overall design strategy (overall design strategy), and progress of the modification process (RQS modification progress), respectively, from the input interface of RQS modification state history maintenance to modification state contents (current RQS modification state contents) the input interface of RQS modification state storage preparation.
- The information link RQS modification state history queries transfers queries on RQS modification states (RQS modification state history queries) from the input interface of RQS modification state history maintenance to the input interface of RQS modification state retrieval.
- The private links persistent RQS modification state information to be stored and persistent RQS modification state information transfer information expressed in information types with the same name from the output interface of RQS modification state storage preparation to the input interface of RQS modification state permanent storage, and from the output interface of RQS modification state permanent storage to the input interface of RQS modification state retrieval, respectively.
- The mediating link temporal RQS modification state information transfers temporal information on RQS modification states (temporal RQS modification state information) from the output interface of RQS modification state storage preparation to the output interface of RQS modification state history maintenance.
- The mediating links RQS modification state history query results, and retrieved persistent RQS modification state information transfer results of queries on RQS modification states (RQS modification state history query results), and persistent information on RQS modification states (persistent RQS modification state information), respectively, from the output interface of RQS modification state retrieval to the output interface RQS modification state history maintenance.

The task control within the component RQS modification state history maintenance is as follows. Upon activation of RQS modification state history maintenance a number of situations are possible, which correspond to task control foci for this component. The task control foci are: initial storage of information, continued storage of information, update of the history, and execute queries. Depending on the specific task control focus, specific links and sub-components are activated, e.g., for update of the history, the component RQS modification state storage preparation is activated and the information links given current RQS name, overall design strategy, and RQS modification progress are made up-to-date. Upon termination of RQS modification state storage preparation, the component RQS modification state permanent storage is activated, and the information links persistent RQS modification state information to be stored and temporal RQS modification state information are made up-to-date. Upon termination of RQS modification state permanent storage, the component RQS modification state history maintenance is terminated.

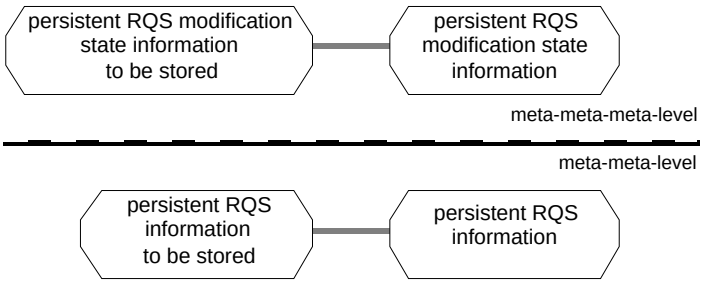


**Figure B.15** Information types related to refinements of sub-processes of RQSM history maintenance.

**Knowledge composition for sub-processes of RQSM history maintenance.** Information types related to refinements of sub-processes of RQSM history maintenance are shown in Figure B.15. The information types persistent RQS information to be stored and persistent RQS modification state information to be stored are different names for persistent RQS information and persistent RQS modification state information, respectively. The persistent information fulfills a different role which is reflected in the name of the information type.

## B.2 Additional refinements of sub-components of DODM

The refinement of several sub-components of DOD manipulation has not been described in Chapter 9. The component DOD modification process co-ordination was introduced in Section 9.3.2 as a sub-component of DOD modification and its composition is described in Section B.2.1. The component assessment point determination was introduced in Section 9.3.5 as a sub-component of DOD modification determination and its composition is described in Section B.2.2. The sub-components of DODM history maintenance have been introduced in Section 9.4.2; their compositions are described in Section B.2.3. Additional knowledge structures related to DODM are described at the end of each of these three sections.

### B.2.1 Refinement of DOD modification process co-ordination

The composition of DOD modification process co-ordination is described by process composition and knowledge composition.

**Process composition for DOD modification process co-ordination: identification of processes and abstraction levels.** The process composition of DOD modification process co-ordination is described by levels of process abstraction, identification of processes, and composition relation between processes.

The process DOD modification process co-ordination is a sub-process of DOD modification, as described in Section 9.3.2. The first level of process abstraction for DOD modification process co-ordination is shown in Figure B.16. The processes DOD modification progress co-ordination, DODM history navigation, DOD modification state analysis, and DOD modification strategy determination are distinguished with the process DOD modification process co-ordination.

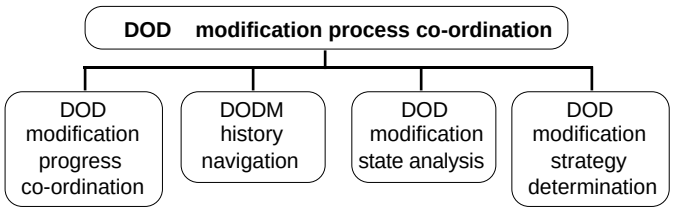


**Figure B.16** Process refinement for DOD modification process co-ordination.

The process DOD modification progress co-ordination is responsible for strategic control within the entire DODM process on the basis of current information on the four sub-processes of DODM. This includes deciding e.g. when to navigate the history, when to validate a specific DOD, etc. The sub-process DODM history navigation is responsible for querying the history component (on DOD modification states, DOD assessment, RQS, and design object descriptions) to find the best suited DOD to continue the modification process, on the basis of available information on global design strategies and manipulation actions. The process DOD modification state analysis analyses a state of the modification process. The process DOD modification strategy determination determines, on the basis of analyses of modification states, which modification strategy is to be employed.

Each of the processes depicted in Figure B.16 can be characterised in terms of their input and output information types, these information types are listed in Table B.4 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| DOD modification progress co-ordination | • overall design strategy description<br>• history navigation results<br>• DOD modification strategy<br>• current modification state contents | • current manipulation action<br>• DOD modification progress<br>• history navigation directives<br>• DOD assessment<br>• DOD modification strategy preferences |
| DODM history navigation | • history navigation directives<br>• DOD modification state history search results<br>• DOD assessment history search results<br>• RQS history search results<br>• DOD history search results<br>• current DOD replacement results<br>• temporal modification state contents analysis | • history navigation results<br>• DOD modification state history queries<br>• DOD assessment history queries<br>• RQS history queries<br>• DOD history queries<br>• current DOD replacement request<br>• historical modification state content |
| DOD modification state analysis | • modification state content | • modification state contents analysis |
| DOD modification strategy determination | • DOD modification strategy preferences<br>• current modification state analysis<br>• historical modification state analysis | • DOD modification strategy |

**Table B.4** Interface information types for sub-processes of DOD modification process co-ordination.

- The process DOD modification progress co-ordination requires the overall design strategy (overall design strategy description), navigation results (history navigation results), DOD modification strategy (DOD modification strategy), and the contents of the current modification state (current modification state contents). This process generates manipulation actions for the overall DODM process (current manipulation action), directives for history navigation (history navigation directives), assessments of design object descriptions (DOD assessment), and preferences on modification strategies (DOD modification strategy preferences).
- The process DODM history navigation needs directives for navigation (history navigation directives), results of searching DOD modification state history ( DOD modification state history search results), results of searching DOD assessment history (DOD assessment history search results), results of searching RQS history (RQS history search results), results of searching DOD history (DOD history search results), results on the success of replacing the current DOD (current DOD replacement results), and analysis of a number of contents of modification states (temporal modification state contents analysis). This process produces navigation results (history navigation results), queries on DOD modification state history ( DOD modification state history queries), queries on DOD assessment history (DOD assessment history queries), queries on RQS history (RQS history queries), queries on DOD history (DOD history queries), requests for replacement of the current DOD (current DOD replacement request), and contents of a (historical) modification state (historical modification state contents).
- The process DOD modification state analysis needs the contents of a modification state (modification state content, i.e., modification foci, current modifications, and assessment of the current DOD). This process generates an analysis of the given modification state contents (modification state contents analysis).
- The process DOD modification strategy determination requires preferences on DOD modification strategies (DOD modification strategy preferences), analysis of the current modification state (modification state analysis), and analysis of a previous DOD modification

state (modification state analysis). This process produces a strategy for the modification of the current DOD (DOD modification strategy).

**Process composition relations within DOD modification process co-ordination.** The information links in the component DOD modification process co-ordination are shown in Figure B.17.

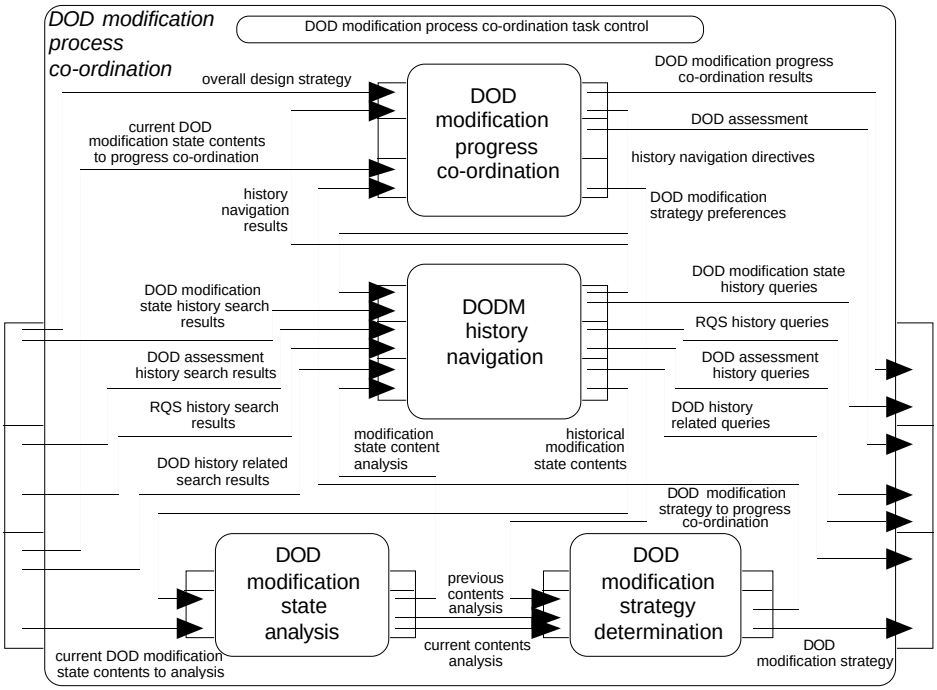


**Figure B.17** Information links within the process of DOD modification process co-ordination.

Within this component fourteen mediating links and eight private links are defined:

- The mediating links current DOD modification state contents to analysis and current DOD modification state contents to progress co-ordination transfer information expressed in modification state contents from the input interface of DOD modification process co-ordination to the input interfaces of DOD modification state analysis and DOD modification progress co-ordination, respectively.
- The mediating link overall design strategy transfers information expressed in overall design strategy from the input interface of DOD modification process co-ordination to the input interface of DOD modification progress co-ordination.
- The mediating links DOD modification state history search results, DOD assessment history search results, RQS history search results, and DOD history related results transfer information expressed in DOD modification state history search results, DOD assessment history search results, RQS history search results, DOD history search results and current DOD replacement results from the input interface of DOD modification process co-ordination to the input interface of DODM history navigation.
- The private links history navigation directives and history navigation results transfer information expressed in history navigation directives and history navigation results, respectively, between DOD modification progress co-ordination and DODM history navigation.
- The private links historical modification state contents and modification state contents analysis transfer information expressed in historical modification state contents and modification state contents analysis, respectively, between DODM history navigation and DOD modification state analysis.
- The private links DOD modification strategy preferences and DOD modification strategy to progress co-ordination transfer information expressed in DOD modification strategy

preferences and DOD modification strategy, respectively, between DOD modification progress co-ordination and DOD modification strategy determination.

- The private links previous contents analysis and current contents analysis transfer information expressed in modification state content analysis to historical modification state analysis and current modification state analysis, respectively, from the output interface of DOD modification state analysis to the input interface of DOD modification strategy determination, on the basis of an explicit mapping between these information types.
- The mediating link DOD modification strategy transfers information expressed in DOD modification strategy from the output interface of DOD modification strategy determination to the output interface of DOD modification process co-ordination.
- The mediating links DOD modification state history queries, RQS history queries, DOD assessment history queries, and DOD history related queries transfer queries and requests to the DODM history from the output interface of DODM history navigation to the output interface of DOD modification process co-ordination.
- The mediating links DOD modification progress co-ordination results, and DOD assessment transfer information expressed in DOD modification progress information and current manipulation action, and DOD assessment, respectively, from the output interface of DOD modification progress co-ordination to the output interface of DOD modification process co-ordination.

The task control within the component DOD modification process co-ordination is as follows. Upon activation of DOD modification process co-ordination, DOD modification progress co-ordination is activated which co-ordinates the sub-processes within DOD modification process co-ordination. A distinction can be made between continuation of the previous manipulation process, or initiation of a new manipulation process. The global phases within DOD modification process co-ordination resemble a process control model. In a process control task a cycle occurs over the sub-tasks: analysis, planning, and execution. Similarly within DOD modification process co-ordination analysis is performed by DOD modification state analysis and DODM history navigation, planning is performed by DOD modification strategy determination and DOD modification progress co-ordination, and execution is performed by effectuating modification strategies, resulting in modifications to the current DOD.

DODM history navigation is able to formulate queries on histories, requests for replacement of the current DOD, and isolate contents of a historical modification state. DOD modification state analysis can be used to analyse the contents of the current modification state, or the contents of a previous modification state (by means of the links current DOD modification state contents and historical modification state contents). DOD modification strategy determination can be activated to determine a modification strategy.

**Knowledge composition for DOD modification process co-ordination.** The information types history navigation directives, history navigation results, modification strategy preferences, DOD modification state contents, historical DOD modification state contents, modification state contents analysis, previous modification state contents analysis, and current modification state contents analysis are described here. Two knowledge bases are described at the end of this section.

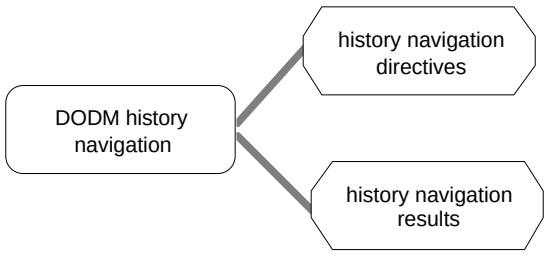


**Figure B.18** Information types related to DODM history navigation.

Two of the information types related to the process DODM history navigation are shown in Figure B.18. The information type history navigation directives contains directives ('goals') for the DODM navigation process. The information type history navigation results contains information on the achievement of the given navigation directives.

Relations in the information type history navigation directives are:

**relations**

| | |
|---|---|
| DOD_to_become_current: | DOD_name; |
| DOD_contents_to_be_retrieved: | DOD_name; |
| comparable_DOD_to_be_retrieved_for: | comparison_type * |
| | DOD_name; |

The relation DOD to become current specifies that the named DOD is to become the current DOD: the contents of the named DOD are to be placed in current DOD maintenance. The relation DOD contents to be retrieved describes which specific DOD has to be retrieved from the DODM history. The relation comparable DOD to be retrieved for specifies that a DOD has to be retrieved which is comparable with the named DOD, e.g., a refinement of the named DOD.

Relations in the information type history navigation results are:

**relations**

| | |
|---|---|
| DOD_contents_retrieved_for: | DOD_name; |
| comparable_DOD_for: | DOD_name * |
| | comparison_type * |
| | DOD_name; |

The relation DOD contents retrieved for specifies that the contents of the named DOD have been retrieved. The relation comparable DOD for specifies that a comparable DOD has been found.

The information type modification strategy preferences is shown in Figure B.19. It contains information on which direction of modification is to be preferred, including, e.g., previously failed modification strategies. Both DOD modification progress co-ordination and DOD modification strategy determination make use of this information type.

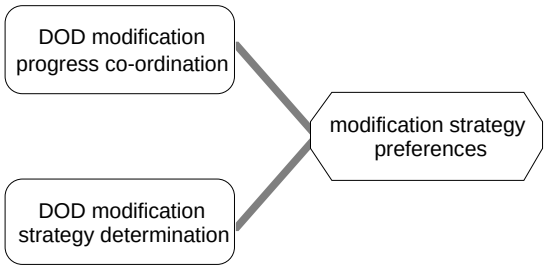


**Figure B.19** The information type modification strategy preferences.

Relations in the information type modification strategy preferences are:

**relations**

| | |
|---|---|
| requirement_qualification_preference: | qualification; |
| rejected_modification_strategy: | modification_method_identification * |
| | modification_method_characterisation; |

The relation requirement qualification preference specifies which requirement qualification is, e.g., to be focussed on. The relation rejected modification strategy specifies which modification strategy is not to be used.

Two information types related to the contents of a modification state are shown in Figure B.20. The information type DOD modification state contents describes current DOD basis evaluation, modification foci, and modifications. The information type historical DOD modification state contents describes the same information. Both information types refer to the same information types, yet their usage in components requires the distinction between contents of *a* modification state, and the contents of a *historical* modification state.

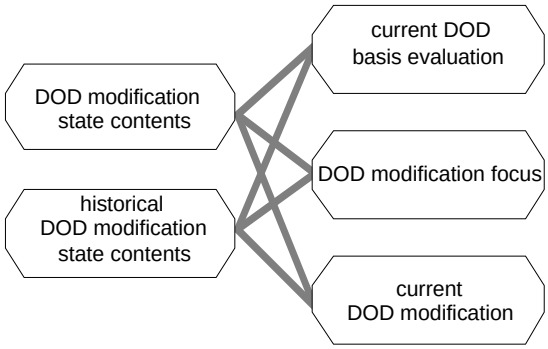


**Figure B.20** Information types related to the contents of a modification state.

The information types related to the analysis of the contents of a modification state are shown in Figure B.21. The information type modification state contents analysis describes the analysis of *a* modification state. The information type temporal modification state contents analysis contains information on the analysis of the contents of a previous and the current modification state. The information type previous modification state contents analysis describes the analysis of a *previous* modification state contents, and the information type current modification state contents analysis describes the analysis of the *current* modification state contents. The process DOD modification state analysis makes use of the information type modification state contents analysis, and the process DODM history navigation makes use of the information type temporal modification state contents analysis.

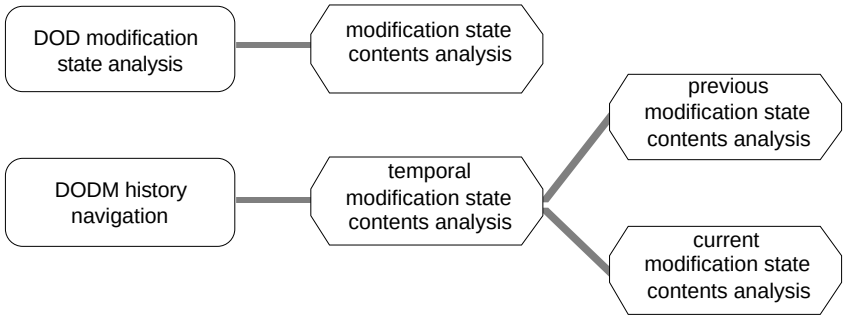


**Figure B.21** Information types related to analysis of the contents of a modification state.

The relation in the information type modification state contents analysis is:

**relations**
content_analysis: DOD_modification_state_content_analysis;

The relation content analysis specifies whether the contents of a modification state contains, e.g., any violated requirements, non-supported qualified requirements with qualification ‘hard’, and non-supported qualified requirements related to an embedded-agent aggregation level.

The relations in the information types previous modification state contents analysis and current modification state contents analysis are:

**relations**
previous_content_analysis: DOD_modification_state_content_analysis;
current_content_analysis: DOD_modification_state_content_analysis;

These two relations specify the analysis of either the contents of a previous modification state, or the current modification state.

Two knowledge bases, related to sub-processes of DOD modification process co-ordination, are shown in Figure B.22. The knowledge base DOD modification state analysis knowledge is employed to analyse the contents of a modification state. An example from this knowledge base is shown below.

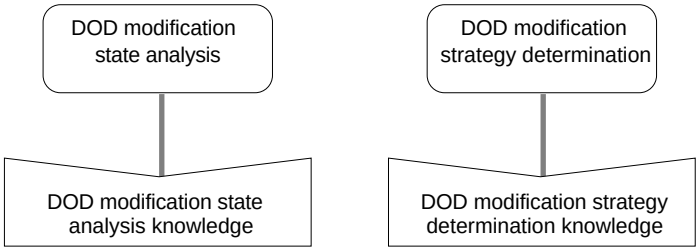


**Figure B.22** Two knowledge bases related to sub-processes of DOD modification process co-ordination.

**Example from knowledge base DOD modification state analysis knowledge**

The knowledge element below illustrates that a non-supported qualified requirement, related to a requirement which has a multi-agent system perspective (i.e., refers to properties related to sets of agents and possibly the external world) leads to the conclusion that the current modification state contains a non-supported qualified requirement with a multi-agent system perspective.

```
if    is_qualified_requirement(                    QR: qualified_requirement_name,
                                                   Q: qualification,
                                                   R: requirement_name )
   and    requirement_has_perspective(   R: requirement_name,
                                         mas_perspective )
   and    not supported_qualified_requirement(   QuRe: qualified_requirement_name )
then      content_analysis(                        contains_non_supported_mas_perspective_qr );
```

The knowledge base DOD modification strategy determination knowledge is employed to determine strategies for the modification process. It is important to note that whenever a DOD is made the current DOD, it is always validated, and after validation a modification strategy is determined (on the basis of the validation results). Examples from this knowledge base are shown below.

**Examples from knowledge base DOD modification strategy determination knowledge**

The knowledge element below illustrates the use of strategy preferences: the preferred requirement qualification is incorporated in the current modification strategy.

```
if   requirement_qualification_preference(   Q: qualification )
   then   DOD_modification_strategy(         qualification_focus,
                                             Q: qualification );
```

The knowledge element below illustrates the use of perspectives on the contents of a DOD: these perspective correspond with aggregation levels for a multi-agent system (Section 5.3). If the analysis of the current modification state shows that qualified requirements are present which are not supported and require properties on the multi-agent system aggregation level, then modification focus identification needs to focus on the mas-perspective.

```
if   current_content_analysis(          contains_non_supported_mas_perspective_qr )
then      DOD_modification_strategy(    perspective_focus,
                                        mas_perspective );
```

### B.2.2 Refinement of assessment point determination

The composition of assessment point determination is described by process composition and knowledge composition.

**Process composition for assessment point determination: identification of processes and abstraction levels.** The process composition of assessment point determination is described by levels of process abstraction, identification of processes, and composition relation between processes.

The process assessment point determination is a sub-process of DOD modification determination, as described in Section 9.3.5. The first level of process abstraction for assessment point determination is shown in Figure B.23. The processes assessment point

derivation, assessment point realisation determination, assessment point expected modification impact determination, and assessment point to realise determination are distinguished within the process assessment point determination.

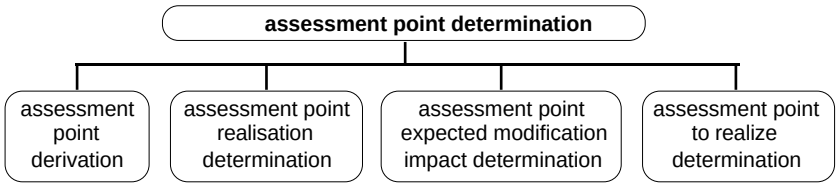


**Figure B.23** Process refinement of assessment point determination.

Identification of assessment points which need to be realised entails four processes. The process assessment point derivation determines which assessment points are of interest, given the current DOD, modification focus, assessments of design requirements, and previously resolved assessment points. The process assessment point realisation determination ascertains which assessment points are already realised. The process assessment point expected modification impact determination categorises assessment points according to their expected impact on the design object description if they were to be realized. On the basis of these expected impacts, an assessment point to be realised is identified by the process assessment point to realise determination.

The interface information types of the sub-processes of assessment point determination are listed in Table B.5 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| assessment point derivation | • current DOD contents<br>• current design requirements<br>• rejected modifications<br>• current DOD basis evaluation<br>• DOD modification focus | • relevant assessment points |
| assessment point realisation determination | • relevant assessment points<br>• current DOD contents | • realised assessment points |
| assessment point expected modification impact determination | • realised assessment points<br>• current DOD basis evaluation<br>• relevant assessment points | • expected modification impact |
| assessment point to realize determination | • selected modification impact<br>• realised assessment points<br>• expected modification impact | • assessment point to be realised |

**Table B.5** Interface information types for sub-processes of assessment point determination

- The process assessment point derivation requires the contents of the current DOD (current DOD contents), design requirements (current design requirements), rejected modifications (rejected modifications), assessment of a DOD on the basis of individual design requirements (current DOD basis evaluation), and a focus for modification (DOD modification focus). This process produces assessment points relevant to the modification foci and assessments of design requirements (relevant assessment points).
- The process assessment point realisation determination requires relevant assessment points (relevant assessment points), and the contents of the current DOD (current DOD contents). This process produces information on the realisation of assessment points (realised assessment points).
- The process assessment point expected modification impact determination needs information on the realisation of assessment points (realised assessment points), assessment of a DOD on the basis of individual design requirements (current DOD basis evaluation), and relevant assessment points (relevant assessment points). This process has as its results an indication of the expected impact when modifications are made to realise non-realised

assessment points (expected modification impact, which is part of the information type modification method evaluation).

- The process assessment point to realise determination needs an indication of the impact of modifications (selected modification impact, which is part of the information type modification method), information on the realisation of assessment points (realised assessment points), and the expected impact of modifications (expected modification impact). As its results it generates assessments points which need to be realised (assessment point to be realised).

**Process composition relations within assessment point determination.** The information links in the component assessment point determination are shown in Figure B.24.

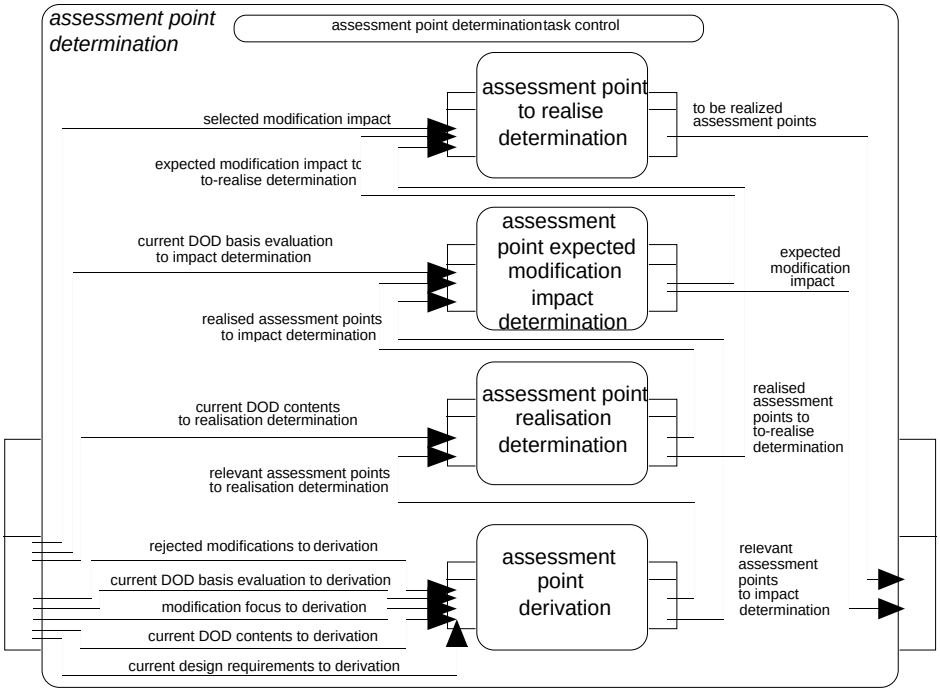


**Figure B.24** Information links within the process of assessment point determination.

Within this component ten mediating links and five private links are defined:

- The mediating link modification focus to derivation transfers modification foci (DOD modification focus) from the input interface of assessment point determination to the input interface of assessment point derivation.
- The mediating link current DOD contents to derivation transfers the contents of the current DOD (current DOD contents) from the input interface of assessment point determination to the input interface of assessment point derivation.
- The mediating link current design requirements to derivation transfers design requirements (current design requirements) from the input interface of assessment point determination to the input interface of assessment point derivation.
- The mediating link current DOD basis evaluation to derivation transfers assessments of a DOD on the basis of individual design requirements (current DOD basis evaluation) from the input interface of assessment point determination to the input interface of assessment point derivation.
- The mediating link rejected modifications to derivation transfers rejected foci for modification (rejected modification focus) from the input interface of assessment point determination to the input interface of assessment point derivation.

- The mediating link current DOD contents to realisation determination transfers the contents of the current DOD (current DOD contents) from the input interface of assessment point determination to the input interface of assessment point realisation determination.
- The mediating link current DOD assessment to impact determination transfers assessments of a DOD on the basis of individual design requirements (current DOD basis evaluation) from the input interface of assessment point determination to the input interface of assessment point expect modification impact determination.
- The mediating link selected modification impact transfers information on the modification method (modification method) from the input interface of assessment point determination to selected impact of modifications (selected modification impact) in the input interface of assessment point to realise determination.
- The private link relevant assessment points to realisation determination transfers assessment points considered to be relevant (relevant assessment points) from the output interface of assessment point derivation to the input interface of assessment point realisation determination.
- The private link relevant assessment points to impact determination transfers assessment points considered to be relevant (relevant assessment points) from the output interface of assessment point derivation to the input interface of assessment point realisation determination.
- The private link realised assessment points to impact determination transfers realised assessment points (realised assessment points) from the output interface of assessment point realisation determination to the input interface of assessment point expected modification impact determination.
- The private link realised assessment points to to-realise determination transfers realised assessment points (realised assessment points) from the output interface of assessment point realisation determination to the input interface of assessment point expected modification impact determination.
- The private link expected modification impact to to-realise determination transfers the expected impact of modifications (expected modification impact) from the output interface of assessment point expected modification impact determination to the input interface of assessment point to realise determination.
- The mediating link to be realised assessment points transfers assessment points to be realised (assessment point to be realised) from the output interface of assessment point to realise determination to the output interface of assessment point determination.
- The mediating link expected modification impact transfers the expected impact of modifications (expected modification impact) from the output interface of assessment point expected modification impact determination to modification method evaluation in the output interface of assessment point determination.

The task control within the component assessment point determination is as follows. Upon activation of assessment point determination a number of possible situations are possible, which correspond to task control foci for this component. The task control foci are: determination of expected modification impact, and determination of assessment points to realise. For both of these task control foci specific task control is needed:

- On activation of task control focus determine expected modification impact, the information links modification focus to derivation, rejected modifications to derivation, current DOD basis evaluation to derivation, current DOD contents to derivation, and current design requirements to derivation are made up-to-date and assessment point derivation is activated. Upon termination of assessment point derivation, assessment point realisation determination is activated, and the information links current DOD contents to realisation determination, and relevant assessment points to realisation determination are made up-to-date. Upon termination of assessment point realisation determination, assessment point expected modification impact determination is activated, and the information links current DOD basis evaluation to impact determination, realised assessment points to impact determination, and

relevant assessment points to impact determination are made up-to-date. Upon termination of assessment point expected modification impact determination, the information link expected modification impact is made up-to-date and assessment point determination terminates itself.

- On activation of task control focus determine assessment points to realise, the information links selected modification impact, expected modification impact to to-realise determination, and realised assessment points to to-realise determination are made up-to-date and assessment point to realise determination is activated. Upon termination of assessment point to realise determination, the information link to be realised assessment points is made up-to-date and assessment point determination terminates itself.

**Knowledge refinement for assessment point determination.** Two information types related to assessment point determination have not been described in Section 9.3.6 and are shown in Figure B.25. The information type relevant assessment points contains information on which assessment points are relevant to the current modification foci. The information type realised assessment points contains information on which assessment point is realised in the current DOD. The information type relevant assessment points is used by the processes assessment point derivation, assessment point realisation determination, and assessment point expected modification impact determination. The information type realised assessment points is used by the processes assessment point realisation determination, assessment point expected modification impact determination, and assessment point to realise determination.

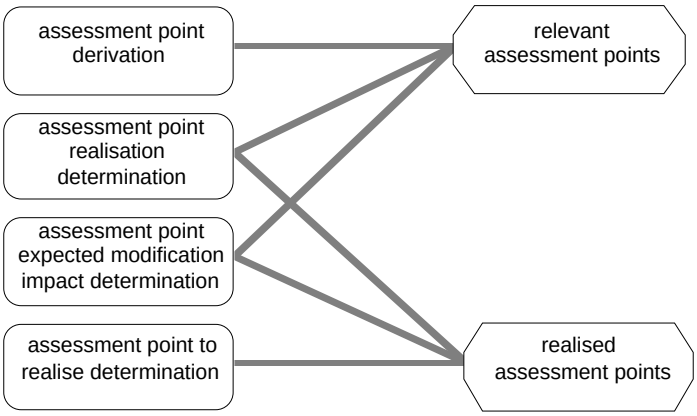


**Figure B.25** Information types related to sub-processes in assessment point determination.

The relation in the information type relevant assessment points is:

```
relations
    relevant:                    design_object_property_atom;
```

The relation relevant specifies which design object property (i.e., assessment point) is relevant (with respect to the current modification focus).

The relation in the information type realised assessment points is:

```
relations
    realised:                    design_object_property_atom;
```

The relation realised specifies which design object property atom (i.e., assessment point) is realised (in the current DOD).

Four knowledge bases, related to sub-processes of assessment point determination, are shown in Figure B.26.

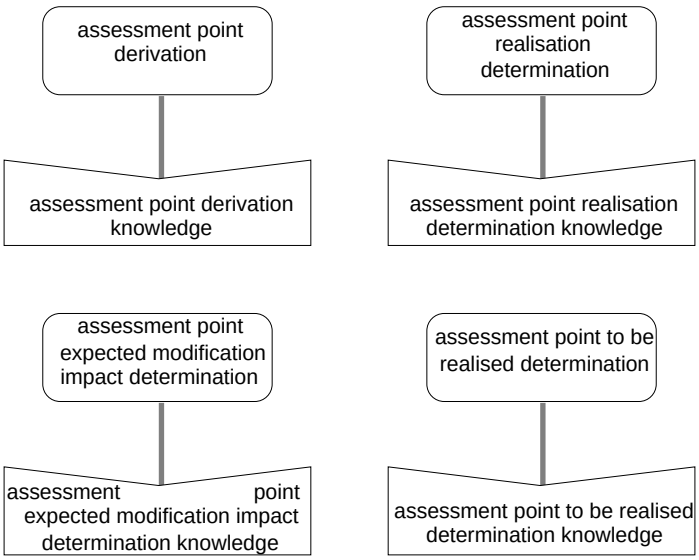


**Figure B.26** Knowledge bases related to sub-processes of assessment point determination.

The knowledge base assessment point derivation knowledge is employed for deriving assessment points for qualified requirements in focus. Example knowledge is shown below.

**Example from knowledge base assessment point derivation knowledge**

The knowledge element below specifies that a qualified requirement in focus, which refers to a requirement with a specific property, is related to a number of assessment points: specific properties which, when fulfilled, realise the required property.

```
if   qr_selected_as_focus( QR: qualified_requirement_name )
     and   is_qualified_requirement( QR: qualified_requirement_name,
                                     Q: qualification,
                                     R: requirement_name )
     and   is_requirement(           R: requirement_name,
                                     has_property( N_D: agent_name,
                                           is_capable_of_reasoning_about_communication_
                                                 from( N_A: agent_name ) ) )
then      assessment_point(
               has_subcomponent_specialised_for(
                         N_D: Name,
                         agent_interaction_management,
                         directed_to( N_A: Name ) ) )
     and   assessment_point(
               has_input_it_specialised_for (
                         N_D: Name,
                         communication_from,
                         agent( N_A: Name ) ) )
     and   assessment_point(
               has_input_it_specialised_for(
                         sub_component_of( agent_interaction_management, N_D: Name ),
                         communication_from,
                         agent( N_A: Name ) ) )
     and   assessment_point(
               has_link_with_contents(
                         N_D: Name,
                         N_D: Name,
                         sub_component_of( agent_interaction_management, N_D: Name ),
                         identity_mapping,
                         communication_from_agent( N_A: Name ),
                         communication_from_agent( N_A: Name ) ) )
```

```
and   assessment_point(
          has_knowledge_base(
                    sub_component_of( agent_interaction_management, N_D: Name ) ) )
and   assessment_point(
          has_knowledge_base_specialised_for(
                    sub_component_of( agent_interaction_management, N_D: Name ),
                    communication_from,
                    agent( N_A: Name ) ) );
```

The knowledge base assessment point realisation determination knowledge is employed to determine which assessment point is already realised by the current DOD. An example from this knowledge base is shown below.

**Example from knowledge base assessment point realisation determination knowledge**

The knowledge element below specifies that an assessment point which holds in the current DOD is considered to be already realised.

```
if    assessment_point( P: design_object_property_atom )
  and holds_in_current_DOD( P: design_object_property_atom, pos )
then  already_realized( P: design_object_property_atom );
```

The knowledge base assessment point expected modification impact determination knowledge is employed to assign an expected modification impact to each non-realised assessment point. An example is shown below.

**Example from knowledge base assessment point expected modification impact determination knowledge**

The knowledge element below specifies that an assessment point, which is not yet realised, has a specific modification impact.

```
if    assessment_point(
          has_input_it_specialised_for(
                    sub_component_of( agent_interaction_management, N_D: Name ),
                    communication_from,
                    agent( N_A: Name ) ) )
then  has_expected_modification_impact(
          has_input_it_specialised_for(
                    sub_component_of( agent_interaction_management, N_D: Name ),
                    communication_from,
                    agent( N_A: Name ) ),
          component_interface_definition );
```

The knowledge base assessment point to be realised determination knowledge is employed to select assessment points which are to be realised, on the basis of a selected modification impact, and expected modification impacts of assessment points.

**Example from knowledge base assessment point to be realised determination knowledge**

The knowledge element below specifies that a non-realised assessment point, with an expected modification impact which is also the selected modification impact is selected to be realised.

```
if    selected_modification_impact(              I: modification_impact )
  and not already_realized(                      P: design_object_property_atom )
  and has_expected_modification_impact(          P: design_object_property_atom,
                                                 I: modification_impact )
then  to_be_realized(                            P: design_object_property_atom );
```

### B.2.3 Additional refinement of DODM history maintenance

The composition of DOD history maintenance, DOD modification results & RQS information history maintenance and DOD modification state history maintenance, sub-processes of DODM history maintenance is described by process composition and knowledge composition.

**Process composition for sub-processes of DODM history maintenance: identification of processes and abstraction levels.** The process compositions for the three sub-processes of DODM history maintenance is described by levels of process abstraction, identification of processes, and composition relation between processes.

The process DODM history maintenance is described in Section 9.4.2. The first level of process abstraction for the sub-processes of DODM history maintenance is shown in Figure B.27. The processes DOD storage preparation, DOD permanent storage, and DOD retrieval are distinguished within the process DOD history maintenance. The processes DOD assessment & RQS storage preparation, DOD assessment & RQS permanent storage, and DOD assessment & RQS retrieval are distinguished with the process DOD assessment & RQS history maintenance. The processes DOD modification state storage preparation, DOD modification state permanent storage, and DOD modification state retrieval are distinguished with the process DOD modification state history maintenance.

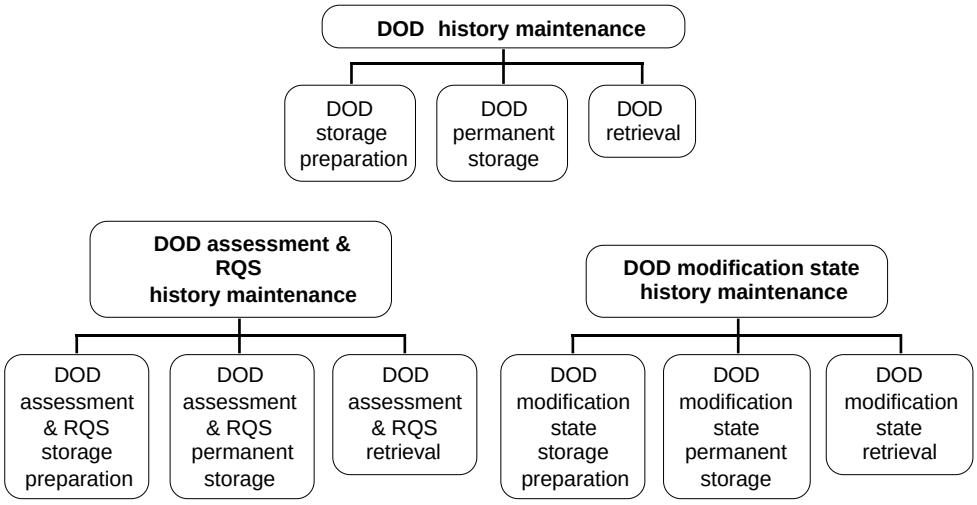


**Figure B.27** Processes at different abstraction levels for DODM History Maintenance.

The process DOD history maintenance is responsible for storing, retrieving and managing descriptions of design objects. Within this process three sub-processes can be distinguished. The process DOD storage preparation assigns names to descriptions of design objects and relates the name of a description to specific description content. The process DOD permanent storage stores the result of the previous sub-process permanently, available for later retrieval. The sub-process DOD retrieval retrieves (parts of) descriptions of design objects according retrieval queries.

The process DOD assessment & RQS history maintenance is responsible for storing, retrieving, and managing DOD assessment and sets of qualified requirements This process is similar to DOD history maintenance. The process DOD assessment & RQS history maintenance consists of three sub-processes. The process DOD assessment & RQS storage preparation is responsible for preparing information to be stored. The process DOD assessment & RQS permanent storage is responsible for storing the prepared information and making it available for the retrieval of parts of DOD assessment or sets of qualified requirements by the process DOD assessment & RQS retrieval, which guides its retrieval on the basis of specific requests for specific information.

The process DOD modification state history maintenance is responsible for storing, retrieving and managing modification states (i.e., information regarding the modification processes). This process is similar to the process DOD history maintenance. The process DOD modification state history maintenance consists of three sub-processes. The process DOD modification state storage preparation is responsible for preparing information to be stored, this entails determining a unique name for the new modification state and relating information on the modification state with that unique name. The process DOD modification state permanent storage

is responsible for storing the resulting DOD modification state and making it available for the retrieval of (parts of) modification states by the process DOD modification state retrieval, which guides its retrieval on the basis of specific requests for specific information.

The interface information types of the sub-processes of DOD history maintenance, DOD assessment & RQS history maintenance and DOD modification state history maintenance are listed in Table B.6 and described below.

| *process* | *input information type* | *output information type* |
|---|---|---|
| *Sub-processes of DOD history maintenance* | | |
| DOD storage preparation | • DOD<br>• current DOD contents | • persistent DOD information to be stored<br>• temporal DOD information<br>• new DOD name |
| DOD permanent storage | • persistent DOD information to be stored | • persistent DOD information |
| DOD retrieval | • persistent DOD information<br>• DOD history queries<br>• current DOD replacement request | • DOD history query results<br>• new current DOD contents<br>• current DOD replacement results<br>• persistent DOD information |
| *Sub-processes of DOD assessment & RQS history maintenance* | | |
| DOD assessment & RQS storage preparation | • RQS<br>• DOD assessment | • DOD assessment & RQS information to be stored |
| DOD assessment & RQS permanent storage | • DOD assessment & RQS information to be stored | • DOD assessment & RQS information |
| DOD assessment & RQS retrieval | • DOD assessment & RQS information<br>• RQS history queries<br>• DOD assessment history queries | • RQS history search results<br>• DOD assessment history search results |
| *Sub-processes of DOD modification state history maintenance* | | |
| DOD modification state storage preparation | • given current DOD name<br>• current DOD modification state contents | • persistent DOD modification state information to be stored<br>• temporal DOD modification state information |
| DOD modification state permanent storage | • persistent DOD modification state information to be stored | • persistent DOD modification state information |
| DOD modification state retrieval | • DOD modification state history queries<br>• persistent DOD modification state information | • DOD modification state history query results<br>• persistent DOD modification state information |

**Table B.6** Interface information types for sub-processes of sub-processes of DODM history maintenance.

The input and output information in the interface of the sub-processes of DOD history maintenance is described below:

- The process DOD storage preparation requires information on descriptions of design objects (DOD) and the contents of the current DOD (current DOD contents). This process produces a name for the current DOD (new DOD name), temporal information on design object descriptions (temporal DOD information), and persistent information on design object descriptions to be stored (persistent DOD information to be stored).
- The process DOD permanent storage needs information on persistent information on design object descriptions to be stored (persistent DOD information to be stored). This process generates persistent information on design object descriptions to be stored (persistent DOD information).

- The process DOD retrieval requires information on persistent information on design object descriptions (persistent DOD information), queries on DOD history (DOD history queries), and requests for replacement of the current DOD (current DOD replacement request). This process produces results of queries on DOD history (DOD history query results), persistent information on design object descriptions (persistent DOD information), results on the success of replacing the current DOD (current DOD replacement results), and new contents for current DOD maintenance (new current DOD contents).

The input and output information in the interface of the sub-processes of DOD assessment & RQS history maintenance is described below:

- The process DOD assessment & RQS storage preparation requires information on sets of qualified requirements (RQS) and assessments of design object descriptions (DOD assessment). This process produces information on sets of qualified requirements and assessments of design object descriptions to be stored (persistent DOD assessment & RQS information to be stored).
- The process DOD assessment & RQS permanent storage needs information on sets of qualified requirements and assessments of design object descriptions to be stored (persistent DOD assessment & RQS information to be stored). This process generates information on sets of qualified requirements and assessments of design object descriptions (persistent DOD assessment & RQS information).
- The process DOD assessment & RQS retrieval requires information on sets of qualified requirements and assessments of design object descriptions (persistent DOD assessment & RQS information), queries on RQS history (RQS history queries), and queries on DOD assessment history (DOD assessment history queries). This process produces results of searching RQS history (RQS information search results), and results of searching DOD assessment history (DOD assessment history search results).

The input and output information in the interface of the sub-processes of DOD modification state history maintenance is described below:

- The process DOD modification state storage preparation requires information on the name given to the current DOD (given current DOD name), contents of the current modification state (current DOD modification state contents). This process produces temporal information on DOD modification states (temporal DOD modification state information), and persistent information on DOD modification states to be stored (persistent DOD modification state information to be stored).
- The process DOD modification state permanent storage needs information on persistent information on DOD modification states to be stored (persistent DOD modification state information to be stored). This process generates persistent information on DOD modification states (persistent DOD modification state information).
- The process DOD modification state retrieval requires information on persistent information on DOD modification states (persistent DOD modification state information), and queries on DOD modification state history (DOD modification state history queries). This process produces results of queries on DOD modification state history (DOD modification state history query results), and persistent information on DOD modification states (persistent DOD modification state information).

**Process composition relations within DOD history maintenance.** The information links in the component DOD history maintenance are shown in Figure B.28.

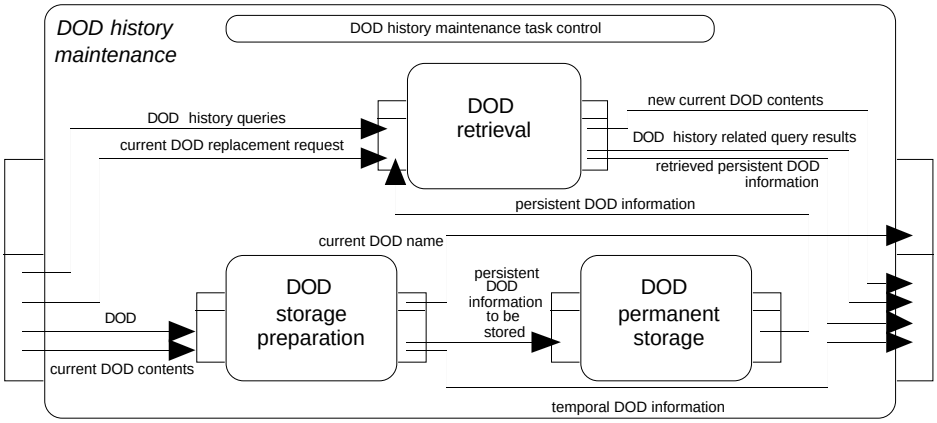


**Figure B.28** Information links within the process of DOD history maintenance.

Within this component nine mediating links and two private links are defined:

- The mediating links DOD and current DOD contents transfer descriptions of design objects (DOD) and the contents of the current DOD (current DOD contents), respectively, from the input interface of DOD history maintenance to the input interface of DOD storage preparation.
- The information links DOD history queries and current DOD replacement request transfer queries on DOD history (DOD history queries) and requests for replacement of the current DOD (current DOD replacement request), respectively, from the input interface of DOD history maintenance to the input interface of DOD retrieval.
- The private links persistent DOD information to be stored and persistent DOD information transfer information expressed in information types with the same name from the output interface of DOD storage preparation to the input interface of DOD permanent storage, and from the output interface of DOD permanent storage to the input interface of DOD retrieval, respectively.
- The mediating link current DOD name transfers the name for the current DOD (new DOD name) from the output interface of DOD storage preparation to information on the current DOD name (current DOD name) the output interface of DOD history maintenance on the basis of an explicit mapping between these information types.
- The mediating link temporal DOD information transfers temporal information on design object descriptions (temporal DOD information) from the output interface of DOD storage preparation to the output interface of DOD history maintenance.
- The mediating links new current DOD contents, DOD history related query results, and retrieved persistent DOD information transfer the new contents of the current DOD (new current DOD contents), results of queries on DOD history (DOD history query results), results on the success of replacing the current DOD (current DOD replacement results), and persistent information on design object descriptions (persistent DOD information), respectively, from the output interface of DOD retrieval to the output interface DOD history maintenance.

The task control within the component DOD history maintenance is as follows. Upon activation of DOD history maintenance a number of situations are possible, which correspond to task control foci for this component. The task control foci are: initial storage of information, continued storage of information, update of the history, execute queries, and replacement of current DOD preparation. Depending on the specific task control focus, specific links and sub-components are activated, e.g., for replacement of current DOD preparation, the component DOD retrieval is activated and the information link current DOD replacement request is made up-to-date. Upon termination of DOD retrieval, the information links new current DOD contents and DOD history related query results are made up-to-date and DOD history maintenance is terminated.

**Process composition relations within DOD assessment & RQS history maintenance.** The information links in the component DOD assessment & RQS history maintenance are shown in Figure B.29.

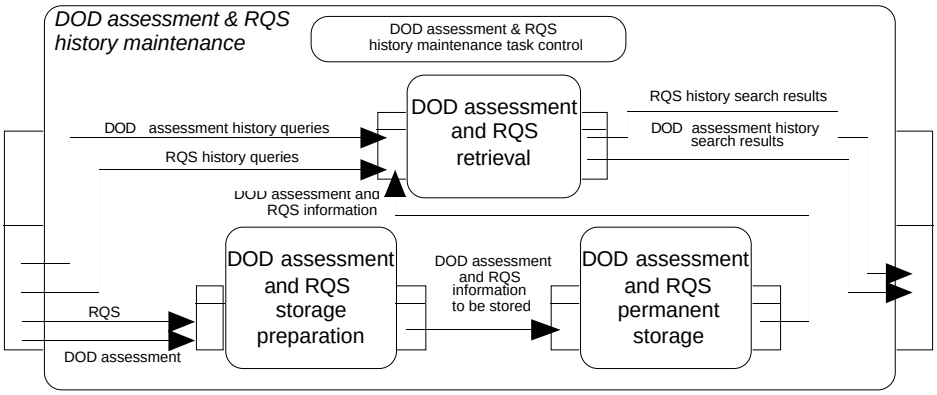


**Figure B.29** Information links within the process DOD assessment & RQS history maintenance.

Within this component six mediating links and two private links are defined:

- The mediating links RQS and DOD assessment transfer sets of qualified requirements (RQS) and assessments of design object descriptions (DOD assessment), respectively, from the input interface of DOD assessment & RQS history maintenance to the input interface of DOD assessment & RQS storage preparation.
- The information links DOD assessment history queries and RQS history queries transfer queries on DOD assessment history (DOD assessment history queries) and queries on RQS history (RQS history queries), respectively, from the input interface of DOD assessment & RQS history maintenance to the input interface of DOD assessment & RQS retrieval.
- The private links DOD assessment & RQS information to be stored and DOD assessment & RQS information transfer information expressed in information types with the same name from the output interface of DOD assessment & RQS storage preparation to the input interface of DOD assessment & RQS permanent storage, and from the output interface of DOD assessment & RQS permanent storage to the input interface of DOD assessment & RQS retrieval, respectively.
- The mediating links RQS history search results, and DOD assessment history search results transfer results on searching the RQS history (RQS history search results), and results of searching DOD assessment history (DOD assessment history search results), respectively, from the output interface of DOD assessment & RQS retrieval to the output interface DOD assessment & RQS history maintenance.

The task control within the component DOD assessment & RQS history maintenance is as follows. Upon activation of DOD assessment & RQS history maintenance a number of situations are possible, which correspond to task control foci for this component. The task control foci are: initial storage of information, continued storage of information, update of the history, and execute queries. Depending on the specific task control focus, specific links and sub-components are activated, e.g., for executing of queries, the component DOD assessment and RQS history retrieval is activated and the information links DOD assessment history queries, RQS history queries, and DOD assessment and RQS information are made up-to-date. Upon termination of DOD assessment & RQS retrieval, the information links RQS history search results, and DOD assessment history search results are made up-to-date and DOD assessment & RQS history maintenance is terminated.

**Process composition relations within DOD modification state history maintenance.** The information links in the component DOD modification state history maintenance are shown in Figure B.30.

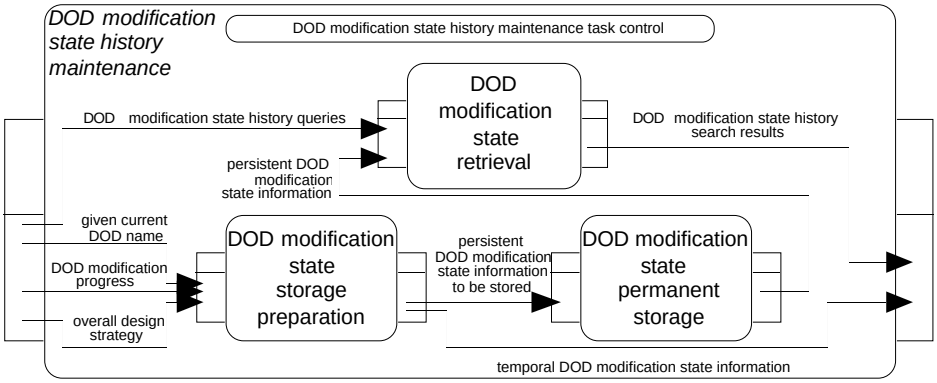


**Figure B.30** Information links within the process of DOD modification state history maintenance.

Within this component six mediating links and two private links are defined:

- The mediating links given current DOD name, overall design strategy, and DOD modification progress transfer the name of the current DOD (given current DOD name), the overall design strategy (overall design strategy), and progress of the modification process (DOD modification progress), respectively, from the input interface of DOD modification state history maintenance to the input interface of DOD modification state storage preparation.
- The information link DOD modification state history queries transfers queries on DOD modification states (DOD modification state history queries) from the input interface of DOD modification state history maintenance to the input interface of DOD modification state retrieval.
- The private links persistent DOD modification state information to be stored and persistent DOD modification state information transfer information expressed in information types with the same name from the output interface of DOD modification state storage preparation to the input interface of DOD modification state permanent storage, and from the output interface of DOD modification state permanent storage to the input interface of DOD modification state retrieval, respectively.
- The mediating link temporal DOD modification state information transfers temporal information on DOD modification states (temporal DOD modification state information) from the output interface of DOD modification state storage preparation to the output interface of DOD modification state history maintenance.
- The mediating link DOD modification state history query results transfers results of searching DOD modification state history (DOD modification state history search results) from the output interface of DOD modification state retrieval to the output interface DOD modification state history maintenance.

The task control within the component DOD modification state history maintenance is as follows. Upon activation of DOD modification state history maintenance a number of situations are possible, which correspond to task control foci for this component. The task control foci are: initial storage of information, continued storage of information, update of the history, and execute queries. Depending on the specific task control focus, specific links and sub-components are activated, e.g., for update of the history, the component DOD modification state storage preparation is activated and the information links given current DOD name, overall design strategy, and DOD modification progress are made up-to-date. Upon termination of DOD modification state storage preparation, the component DOD modification state permanent storage is activated, and the information links persistent DOD modification state information to be stored and temporal DOD modification state information are made up-to-date. Upon termination of DOD modification state permanent storage, the component DOD modification state history maintenance is terminated.

**Knowledge composition for sub-processes of DODM history maintenance.** Information types related to refinements of sub-processes of DODM history maintenance are

shown in Figure B.31. The information types persistent DOD information to be stored, and persistent DOD modification state information to be stored are different names for persistent DOD information, and persistent DOD modification state information, respectively. The persistent information fulfills a different role which is reflected in the name of the information type. The information types DOD assessment & RQS information to be stored and DOD assessment & RQS information refer to the same information types: DOD assessment history and persistent RQS information. Their names differ, reflecting the difference in usage of the information types.

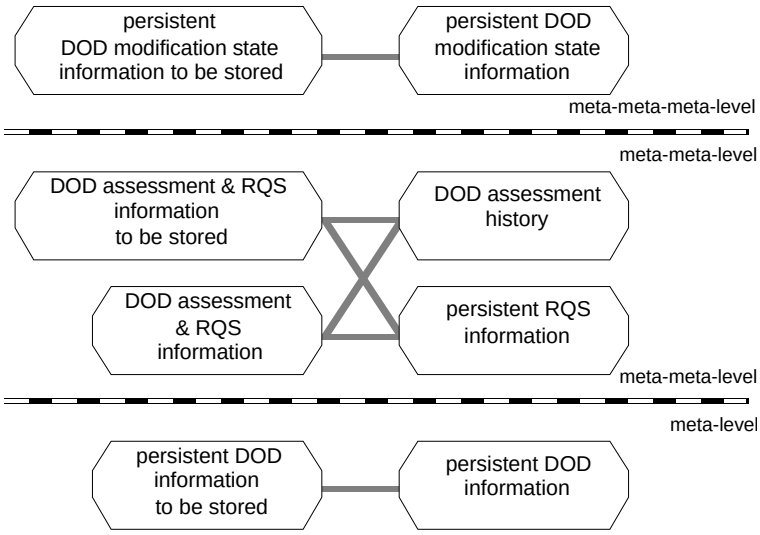


**Figure B.31** Information types related to refinements of sub-processes of DODM history maintenance.

# Summary

## Re-design of compositional systems

The central research theme of this thesis is: *How can a compositional structure be used to re-design knowledge-intensive systems?*

*Design* is an activity common to humans; a fact testified by the presence of the many artefacts in our surroundings. Often new artefacts are designed on the basis of existing artefacts, an activity named re-design. Design can be viewed as a process of the creation of a set of requirements and a design object description that satisfies these requirements, on the basis of initial requirements and preferences specified by agents, and libraries of existing designs, while adhering to design process objectives.

Requirements guide the direction in which solutions are sought and determine which properties will be used to evaluate the results of the process. In this thesis requirements on compositional systems are formulated in terms of required properties. Such properties refer to static aspects of a compositional system (i.e., the structure), or dynamic aspects of a compositional system (i.e., the behaviour), or both.

The approach of identifying entities called components and defining a way to combine components together to make a new component, is termed *compositional design*. Compositionality can be applied to both *processes* and *knowledge* within one compositional system. A compositional structure can be used not only to structure the (software) design object but also to structure the process of re-design (a knowledge-intensive process by itself).

The compositional approach to design may be applied in many different domains of application. The two domains of application addressed in this thesis are diagnostic reasoning systems and self-modifying multi-agent systems.

*Diagnostic reasoning systems* are systems in which a faulty component of an artefact is detected on the basis of observations on the behaviour of the artefact. These observations can be given beforehand, or can be made 'when necessary'.

The (multi-) agent paradigm provides a means to characterise autonomous distributed processes. The systems (either human or automated) responsible for these processes are the agents in the multi-agent system. Each agent has its own environment, consisting of other agents and a world. Agents are able to communicate with each other, can co-operate to jointly perform tasks, interact with the world (observe and/or act), and perform specific tasks. The agent metaphor can also be used to develop agents that are able to dynamically design and create new agents, or to dynamically modify existing agents. The domain of *self-modifying multi-agent systems* is a rich domain of application for re-design. It provides a natural setting for a process of re-design: an existing multi-agent system is re-designed by one (or more) of its agents.

This thesis uses an existing generic design model (GDM) as the basis for design: a model in which manipulation of design object descriptions, manipulation of sets of qualified requirements, and co-ordination of the design process are the three main processes distinguished. The model GDM is refined into a *model for re-design* of compositional systems.

This model for re-design of compositional systems has been applied to the re-design of a compositional knowledge-based diagnostic reasoning system. The model for re-design of compositional systems has also been combined with the generic agent model GAM, to acquire a model for a *design agent*. The design agent plays an important role in a self-modifying multi-agent system which has been modelled, specified and implemented using the DESIRE environment. The design agent designs and dynamically adds, at runtime, a new agent to the existing multi-agent system. The trace of the re-design process illustrates the integration of the result of a re-design process: a design object (artefact) is implemented according to the resulting

description. A realisation action based on the generated design modifies the multi-agent system itself, thereby effectuating self-modification of the system.

The research presented in this thesis answers the central research theme, namely “How can a compositional structure be used to re-design knowledge-intensive systems?”. The applications of the model for re-design of compositional systems have illustrated how a compositional structure can be used: compositionality as a structuring principle for describing design processes, and compositionality as a structuring principle for describing design object descriptions. The application of the model of re-design of compositional systems to self-modifying (multi-agent) systems is a step towards flexible, adaptive systems.

# Samenvatting

## Herontwerp van compositionele systemen

De onderzoeksvraag van dit proefschrift luidt: *hoe kan een compositionele structuur gebruikt worden voor het herontwepren van kennis-intensieve systemen?*

*Ontwerpen* is een activiteit die mensen eigen is, zoals de aanwezigheid van vele artefacten in onze omgeving laat zien. Er is sprake van herontwerpen wanneer nieuwe artefacten ontworpen worden aan de hand van een bestaand artefact. Ontwerpen kan gezien worden als een proces waarbij een verzameling eisen wordt geformuleerd en een beschrijving van een ontwerpobject wordt geproduceerd dat hieraan voldoet. Leidraad daarbij zijn initiële eisen en voorkeuren zoals aangegeven door agenten en bibliotheken van bestaande ontwerpen, terwijl het proces beantwoordt aan de doelen die gesteld zijn aan het ontwerpproces.

In een ontwerpproces spelen eisen een belangrijke rol: ze sturen de richting waarin oplossingen worden gezocht en bepalen op grond van welke eigenschappen het resultaat van het proces wordt geëvalueerd. Eisen aan compositionele systemen worden in dit proefschrift uitgedrukt in termen van vereiste eigenschappen. Eigenschappen refereren aan statische (bijvoorbeeld de structuur), of dynamische aspecten (bijvoorbeeld het gedrag) van een compositioneel systeem, of aan beide.

De aanpak waarbij componenten worden onderscheiden en waarbij een manier gezocht wordt om componenten te combineren tot nieuwe componenten, wordt *compositioneel ontwerpen* genoemd. Binnen een compositioneel systeem kunnen zowel processen als kennis compositioneel beschreven worden. Zo'n compositionele structuur kan niet alleen gebruikt worden om het (software) ontwerpobject te structureren, maar ook om het proces van herontwerp (dat zelf een kennis-intensief proces is) te structureren.

De compositionele aanpak bij het ontwerpen van compositionele systemen is toegepast in een aantal verschillende toepassingsgebieden. In dit proefschrift worden de toepassingsgebieden diagnostische redeneersystemen en multi-agent systemen gebruikt.

*Diagnostische redeneersystemen* zijn systemen waarin een foutieve component van een artefact wordt gevonden aan de hand van observaties van het gedrag van het artefact. Observaties kunnen van te voren gegeven zijn, of ze kunnen gedaan worden 'wanneer nodig'.

Het (multi-)agent paradigma maakt het mogelijk om autonome, gedistribueerde processen te karakteriseren. De (menselijke of geautomatiseerde) systemen die verantwoordelijk zijn voor deze processen, zijn de agenten in het multi-agent systeem. Elke agent heeft zijn eigen omgeving, bestaande uit andere agenten en een wereld. Agenten kunnen met elkaar communiceren, samenwerken om gezamenlijk een taak uit te voeren, interactie vertonen met de wereld (observeren en/of handelen) en specifieke taken uitvoeren. De agent-metafoor kan ook worden gebruikt om agenten te bouwen die zelf dynamisch nieuwe agenten kunnen ontwerpen en creëren, danwel dynamisch bestaande agenten aanpassen. Deze zogenaamde *zelf-modificerende multi-agent systemen* bieden veel toepassingen. Ze vormen een natuurlijke inbedding voor een herontwerpproces: een bestaand multi-agent systeem wordt herontworpen door een (of meerdere) van de agenten in het systeem zelf.

In dit proefschrift wordt een bestaand, generiek model van ontwerp (GDM, generic design model) gebruikt waarin drie deelprocessen worden onderscheiden: het manipuleren van het ontwerpobject, het manipuleren van verzamelingen van gekwalificeerde eisen en het coördineren van het ontwerpproces. Het model GDM is verfijnd in een *model voor herontwerpen* van compositionele systemen.

Het herontwerpmodel voor compositionele systemen is toegepast op een compositioneel kennis-gebaseerd diagnostisch redeneersysteem. Het herontwerpmodel voor compositionele systemen is gecombineerd met het generieke agent model GAM; dit heeft een *model voor een*

*ontwerpagent* opgeleverd. De ontwerpagent speelt een belangrijke rol in een zelf-modificerend multi-agent systeem, en is gemodelleerd, gespecificeerd en geïmplementeerd op basis van de DESIRE omgeving. De ontwerpagent voegt op dynamische wijze een zelfontworpen agent toe aan het bestaande in werking zijnde multi-agent systeem. De stappen die in dit proces zijn doorlopen laten zien hoe een ontwerpobject wordt geïmplementeerd aan de hand van de zelfontworpen beschrijving van een agent. Een 'realisatie-actie' op grond van een ontwerp verandert het eigenlijke multi-agent systeem, waarmee zelf-modificatie van een systeem werkelijkheid wordt.

Met het onderzoek zoals beschreven in dit proefschrift, is de onderzoeksvraag "hoe kan een compositionele structuur gebruikt worden voor het herontwerpen van kennis-intensieve systemen?" beantwoord. De toepassingen van het herontwerpmodel voor compositionele systemen geven aan hoe een compositionele structuur benut kan worden: als structureringsprincipe voor het beschrijven van ontwerpprocessen en als structureringsprincipe voor het beschrijven van het ontwerpobject. De toepassing van het herontwerpmodel voor compositionele systemen in zelf-modificerende (multi-agent) systemen is een stap op weg naar flexibele, zichzelf aanpassende, systemen.

# SIKS dissertatiereeks

## 1998

98-1 Johan van den Akker (CWI)
*DEGAS - An Active, Temporal Database of Autonomous Objects*
Promotor: prof.dr. M.L. Kersten (CWI/UvA)
Co-promotor: dr. A.P.J.M. Siebes (CWI)
Promotie: 30 maart 1998

98-2 Floris Wiesman (UM)
*Information Retrieval by Graphically Browsing Meta-Information*
Promotores: prof.dr.ir. A. Hasman (UM)
prof.dr. H.J. van den Herik (UM/RUL)
prof.dr.ir. J.L.G. Dietz (TUD)
Promotie: 7 mei 1998

98-3 Ans Steuten (TUD)
*A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*
Promotores: prof.dr.ir. J.L.G. Dietz (TUD)
prof.dr. P.C. Hengeveld (UvA)
Promotie: 22 juni 1998

98-4 Dennis Breuker (UM)
*Memory versus Search in Games*
Promotor: prof.dr. H.J. van den Herik (UM/RUL)
Promotie: 16 oktober 1998

98-5 E.W.Oskamp (RUL)
*Computerondersteuning bij Straftoemeting*
Promotores: prof.mr. H. Franken
prof.dr. H.J. van den Herik
Promotie: 13 mei 1998

## 1999

99-1 Mark Sloof (VU)
*Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products*
Promotor: prof.dr. J. Treur
Co-promotor: dr.ir. M. Willems
Promotie: 11 mei 1999

99-2 Rob Potharst (EUR)
*Classification using decision trees and neural nets*
Promotor: prof.dr. A. de Bruin
Co-promotor: dr. J.C. Bioch
Promotie: 4 juni 1999

99-3 Don Beal (Queen Mary and Westfield College)
*The Nature of Minimax Search*
Promotor: prof.dr. H.J.van den Herik
Promotie: 11 juni 1999

99-4 Jacques Penders (KPN Research)
*The practical Art of Moving Physical Objects*
Promotor: prof.dr. H.J. van den Herik
Co-promotor: dr. P.J. Braspenning
Promotie: 11 juni 1999

99-5 Aldo de Moor (KUB)
*Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*
Promotor: prof.dr. R.A. Meersman
Co-promotor: dr. H. Weigand
Promotie: 1 oktober 1999

99-6 Niek J.E. Wijngaards (VU)
*Re-design of compositional systems*
Promotor: prof.dr. J. Treur
Co-promotor: dr. F.M.T. Brazier
Promotie: 30 september 1999